A Hebbian Recurrent Neural Network Explains the Hierarchical Geometry of Sequence Memory

Zhitao FENG[1]
xy1204655650@gmail.com

Bo Hong[2]
hongbo@tsinghua.edu.cn

Huan Luo[3,✉]
huan.luo@pku.edu.cn

Xiaolong Zou[2,✉]
benzou@mail.bnu.edu.cn

Yuanyuan Mi[1,✉]
miyuanyuan@tsinghua.edu.cn
1, Department of Psychological and Cognitive Sciences, Tsinghua University.
2, School of Biomedical Engineering, Tsinghua Medicine, Tsinghua University.
3, School of Psychological and Cognitive Sciences, Peking University.
✉: Corresponding authors.

Abstract

Temporal sequences in working memory are hierarchically organized to support flexible behavior. Recent neural recordings suggest that this organization relies on a 1D-to-2D neural geometrical folding in working memory, whereby ordinal positions are re-encoded within an orthogonal scaffold, with two dimensions indexing global (chunk-level rank) and local ranks (within-chunk position), respectively. These findings implicate an important structural principle for memory, yet the computational mechanisms underlying this geometry remain unclear. Here, we study this question using a recurrent neural network with fast Hebbian plasticity (Hebb-RNN). When meta-trained on rank recalling tasks, the Hebb-RNN generalizes to novel item-rank bindings and hierarchical sequences, outperforming control models. Mechanistic analysis reveals that the network develops an approximately orthogonal ordinal geometry consistent with empirical findings. This geometry transforms temporal inputs into abstract ordinal ranks that provide a memory scaffold, while Hebbian plasticity supports rapid conjunctive binding of item content to this structure. Moreover, the pre-learned memory folding process accelerates learning in novel tasks, supporting efficient transfer. Finally, our model extends to naturalistic encodings and acquires hierarchical geometry without externally provided chunking signals. Together, our results provide a biologically plausible computational framework for hierarchical sequence representation, offering possible new insights into structured memory organization for machine learning.

1 Introduction

Temporal sequences in memory are not stored passively but are actively organized into structured units [1]. A familiar example is phone number memorization, where digits are grouped into chunks rather than encoded in isolation [2]. Grouping contiguous elements into higher-level units while preserving their internal order reflects a general principle of memory organization [3–8]. This hierarchical structuring is observed across cognitive domains, from narrative segmentation during reading [9] to action chunking and compression in motor behavior [10]. It is thought to alleviate capacity constraints and support flexible behavior [11, 12, 2, 13–15]. Despite its central role in high-

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

level cognition, how temporal sequences are hierarchically represented and organized in memory remains unclear.

Recently, an elegant study by Fan et al. demonstrated that sequential order in working memory is represented using an orthogonal coding scheme [16]. In a rank-recalling task, neural recordings revealed that the brain spontaneously chunks syllable sequences into multisyllabic words, which are then folded into an approximately two-dimensional neural geometry: a global dimension indexing chunk-level rank and a local dimension indexing within-chunk position (Fig. 1A). This geometric framework provides a unifying explanation for earlier behavioral and single-unit findings [17, 18], naturally accounting for the systematic positional confusions observed between items that share the same within-chunk position across different chunks. Similar 2D neural geometry also appears in episodic memory. In rats, activity in the lateral entorhinal cortex organizes experience along two approximately orthogonal dimensions, capturing both episode progression and within-episode order [19, 20].

Together, these findings suggest that 2D neural geometry is a fundamental principle for organizing temporal sequences in memory, enabling local circuits to represent multiscale temporal structure simultaneously. However, how this geometry emerges and supports the 1D-to-2D neural geometric folding in memory remains largely unknown.

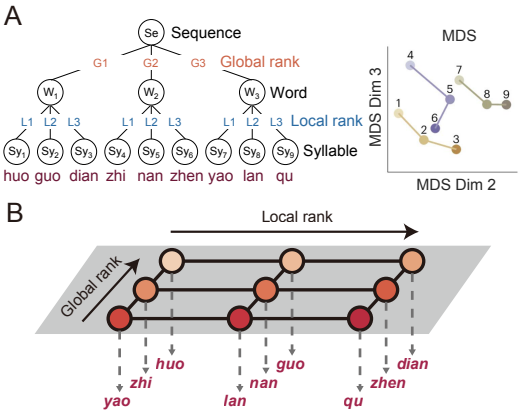

Figure 1: 2D Neural Geometry in Working Memory and Our Hypothesis. (A) Neural geometry in rank-recalling memory tasks, adapted from [16]. When a sequence of syllables with hierarchical structure is presented (upper panel), neural recordings reveal that the temporally 1D sequence is embedded within an approximately 2D neural geometry. (lower panel). (B) 2D memory scaffold hypothesis. An orthogonal rank geometry, composed of two mutually orthogonal rank dimensions and factorized from memory content, dynamically organizes sequence items.

Early modeling work proposed several classes of sequence models, including ordinal [21] and positional models [22, 23]. In particular, positional models assume that each item is associated with a positional code, such that recall is guided by ordinal signals rather than item-to-item associations. Despite their conceptual appeal, these frameworks lacked concrete neural implementations. More recently, a range of hippocampus-inspired sequence models have been proposed [24–26]. Notably, James Whittington et al. introduced a working memory model in which a velocity-driven recurrent system manipulates structured activity slots to encode ordinal sequences [26], accounting for diverse memory phenomena. However, these models primarily address 1D sequences without hierarchical structure and rely on a predefined factorized structure-content prior.

We propose that an abstract, content-independent 2D ordinal geometry - defined by orthogonal global and local rank axes - serves as a scaffold for organizing hierarchical memory sequences (Fig. 1B), and we test this hypothesis using a recurrent neural network with fast Hebbian plasticity, termed Hebb-RNN. Our Hebb-RNN generalizes compositionally to novel item-rank bindings and hierarchical structures, substantially outperforming control models, while an approximately orthogonal 2D neural geometry spontaneously emerges, offering a mechanistic account of human rank coding. We further demonstrate that memory sequences undergo a 1D-to-2D geometric folding in the network, constructing a scaffold where global and local ranks are computed and bound to items via fast Hebbian plasticity; this learned geometry and folding process also accelerates new task learning. In naturalistic settings, the Hebb-RNN can derive chunk structure from input statistics without externally provided chunking signals, supporting compositional rank generalization over real-word encodings while maintaining an approximately 2D orthogonal geometry. Together, our findings provide a framework for understanding the emergence of multiscale geometric representations of sequence memory and may provide new insights for memory organization in machine learning.

2 Methods

2.1 Network Structure and Synaptic Plasticity

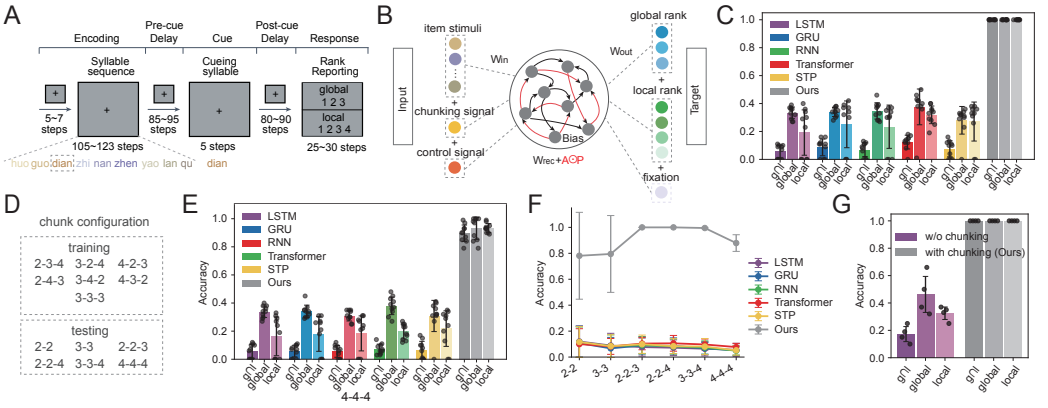

Figure 2: Modeling Rank Recalling Tasks: Hebb-RNN Architecture and Generalization Performance. (A) Task schematic. The task has five phases: encoding, pre-cue delay, cue, post-cue delay, and response. During encoding, an example 3-3-3 sequence is presented. After a pre-cue delay, a cue stimulus is given, and the network predicts the corresponding local and global ranks following the post-cue delay. (B) Model architecture. Hebbian plasticity is implemented in the recurrent weights of the RNN. Inputs include a stimulus vector, a 1D control signal, and a chunking signal. The network outputs vectors representing local and global rank predictions, as well as a fixation signal related to task behavior. (C) Within-distribution test accuracy. Models are trained and tested on the same sequence structures listed in the upper panel in (D). Hebb-RNN performance is compared with LSTM, GRU,vanilla RNN, 3-layer transformer and RNN with short-term plasticity (STP) [27]. All models receive the same inputs and output as the Hebb-RNN. $g \cap l$ denotes accuracy when both local and global predictions are correct; "local" and "global" denote accuracy for the respective predictions. (D) Out-of-distribution sequence structures. Models are trained on 7 sequence configurations (e.g., 2-3-4: three chunks of 2, 3, and 4 items) and tested on other different, non-overlapping configurations (e.g., 4-4-4). (E) Out-of-distribution test accuracy. Test performance of different architectures on the 4-4-4 sequence configuration. (F) Generalization across sequence structures. (G) Model performance of Hebb-RNNs with and without external chunking signals. All test accuracies are averaged over 10 seeds. Detailed method is provided in Appendix A.3.

The Hebb-RNN is a simple recurrent neural network endowed with fast Hebbian plasticity (Fig. 2B), inspired by the proposed role of rapid synaptic plasticity in flexible binding and transient memory storage [28–30]. It follows a standard input-hidden-output architecture. At each time step, the network receives three types of inputs: item stimuli forming a sequence, a chunking signal indicating sequence segmentation, and a control signal cueing response timing. In the brain, the chunking signal can be driven by prediction errors or novelty [20, 7, 8, 31, 4, 32], while the control signal regulates behavioral responses. The output layer consists of three components: a global-rank output vector, a local-rank output vector, and a fixation output.

At each discrete time step t, Hebb-RNN receives an input vector $I(t) = [I_s(t); I_g(t); I_c(t)]$, where $I_s(t) \in \mathbb{R}^{N_s}$ denotes the stimulus vector, and $I_g(t)$ and $I_c(t)$ are scalar signals corresponding to the sequence chunking and control signal, respectively. The hidden-state dynamics is described as below,

$$\tau \frac{d\mathbf{h}(t)}{dt} = -\mathbf{h}(t) + (W_{rec} + A \odot P(t))\mathbf{r}(t) + W_{in} I(t) + \mathbf{b}, \tag{1}$$

where τ is the time constant. Here, $W_{in} \in \mathbb{R}^{N_{hid} \times N_{in}}$ and $W_{rec} \in \mathbb{R}^{N_{hid} \times N_{hid}}$ denote the input and recurrent weight matrices, respectively. The vector $\mathbf{b}$ is a bias term. The hidden activity is transformed via $\mathbf{r}(t) = \tanh(\mathbf{h}(t))$. The continuous-time formation is discretized using the Euler method.

To model synaptic plasticity, the recurrent connectivity includes a dynamic component $P(t) \in \mathbb{R}^{N_{hid} \times N_{hid}}$. The matrix $A \in \mathbb{R}^{N_{hid} \times N_{hid}}$ gates plastic contributions through element-wise multipli-

cation. Inspired by biological synaptic plasticity, the matrix $P(t)$ is updated online using a Hebbian learning rule at time t,

$$P(t+1) = P(t) + \eta \mathbf{h}(t)\mathbf{r}(t-1)^T, \tag{2}$$

where η denotes the plasticity learning rate. Given a readout matrix $W_{out} \in \mathbb{R}^{N_{out} \times N_{hid}}$, the network output at time t is $\mathbf{y}(t) = W_{out}\mathbf{r}(t)$.

Let the target output sequence be denoted by $\hat{\mathbf{y}}(t) \in \mathbb{R}^{N_{\text{out}}}$. At each time step, the network predicts an output $\mathbf{y}(t)$ and is trained by minimizing a mean squared error loss,

$$L = \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} |\mathbf{y}_b(t) - \hat{\mathbf{y}}_b(t)|_2^2 + \frac{\alpha}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} |\mathbf{r}_b(t)|_2^2 , \tag{3}$$

where B denotes the batch size and T the number of time steps. The second term acts as a regularization penalty on hidden neural activity, encouraging low-activity solutions and mimicking energy-efficient coding in biological neural systems. α controls the regularization strength. See Appendix A.2 for more details.

The network is optimized using backpropagation through time and meta-trained on a variety of rank-recall memory tasks, in which sequence items are randomly sampled but share the same underlying rank structures (see Fig. 2D). The parameters $[W_{in}, W_{rec}, \mathbf{b}, W_{out}, A]$ are trainable and fixed after training, while the plastic matrix $P(t)$ evolves online according to a Hebbian update rule and is reset to zero at the beginning of each trial. Through meta-training, the Hebb-RNN learns to exploit fast synaptic plasticity to support memory encoding and recall. Full details of hyperparameters, training procedures, optimizers, and loss functions are provided in Appendix A.2.

2.2 A Rank Recalling Working Memory Task

Following Fan et al. [16], we evaluate our Hebb-RNN using a rank-recalling working-memory task (Fig. 2A). The task consists of five stages: an encoding phase, a pre-cue delay, a cue period, a post-cue delay, and a response phase. During encoding, a hierarchical sequence of items is presented sequentially. After a brief pre-cue delay, one item is randomly sampled from the sequence and presented as a cue, followed by a post-cue delay, after which the network is required to report both the global (chunk-level) and local (within-chunk) ranks associated with the cued item. The presented sequences have explicit hierarchical structure. For example, the sequence shown in Fig. 2A follows a 3-3-3 organization, consisting of three chunks with three items each. To focus on underlying computational mechanisms, individual items are simplified and represented using one-hot vectors here; alternatively, random vectors sampled from a Gaussian distribution are also used (see Appendix A.4.6). Motivated by biological systems, where prediction errors from sensory or subcortical regions may trigger chunk-boundary signals to the working memory system [20, 7, 8, 31, 4, 32], we introduce an explicit chunking signal that is transiently activated at chunk boundaries during encoding. To assess generalization, we evaluate models under both within-distribution and out-of-distribution conditions. Further task details are provided in Appendix A.1.

3 Results

3.1 Hebb-RNNs Generalize Across Diverse Task Settings

To evaluate the effectiveness of our model, we train and test Hebb-RNNs and diverse control recurrent architectures on a rank-recalling working memory task (Fig. 2A). In this task, memory sequences exhibit a hierarchical rank structure, comprising both within-chunk position and chunk-level rank. Note that although we use toy stimuli, each item can appear at arbitrary global and local rank positions, making the task nontrivial: it requires compositional generalization, not simple memorization.

We first assess generalization in a within-distribution setting, in which models are trained and evaluated on the same sequence structures (e.g., 3-3-3, 2-3-4, etc.).During testing, the test set is strictly disjoint from the training set, ensuring evaluation on previously unseen item-rank combinations. Shown in Fig. 2C, Hebb-RNNs generalize perfectly, accurately predicting both global and local ranks for all novel combinations. In contrast, LSTM, GRU, vanilla RNN, 3-layer Transformer, and RNN with short-term plasticity [27] perform at chance level under the matched training setting ($g \cap l$ bars in Fig. 2C) and remain well below the Hebb-RNN despite increased model capacity and

extended training (Appendix A.4.1). The Transformer can match the Hebb-RNN's performance with substantially more training, reaching $94.8\% \pm 10.8\%$ accuracy with 10,000 sequences and $100.0\% \pm 0.0\%$ with 30,000 sequences after 150 epochs, compared with $100.0\% \pm 0.0\%$ for the Hebb-RNN using only 3,000 sequences and 50 epochs, highlighting the superior sample and training efficiency of the Hebb-RNN.

We next evaluate a more stringent form of generalization in an out-of-distribution setting. In this setting, we test models under unseen structure configurations not present during training (Fig. 2D). Shown in Fig. 2E, when tested on a 4-4-4 configuration - whose total sequence length exceeds that of any sequence encountered during training - the Hebb-RNN still maintains good performance, achieving approximately 90% accuracy. In contrast, all control recurrent models fail to generalize to this unseen structure. Broad generalization across a wide range of previously unseen sequence configurations is further demonstrated in Fig. 2F and Appendix A.4.4.

Additionally, Hebb-RNNs trained on Gaussian stimuli and diverse plasticity rules also perform well (Appendix A.4.6). Together, Hebb-RNNs achieve good compositional generalization, generalizing to both novel item-rank combinations and new sequence structures in a rank-recalling task.

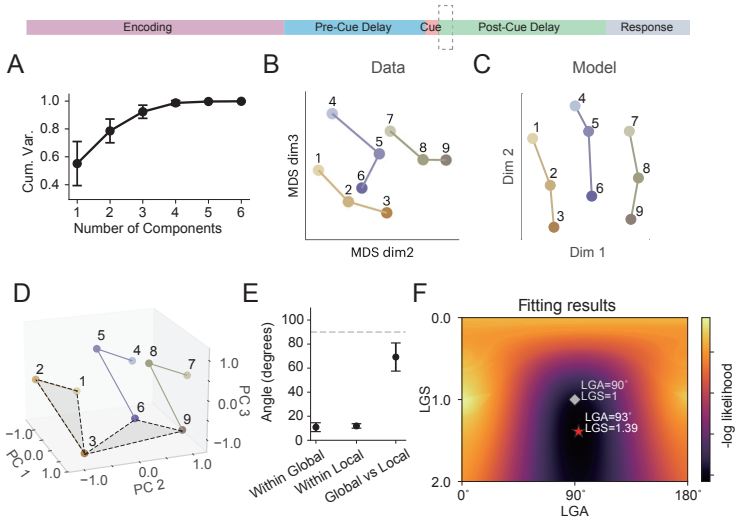

Figure 3: Hierarchical Geometry of Temporal Rank Representations in a Hebb-RNN. Following Fan et al.'s work [16], we analyze the neural geometry of rank representations during the early maintenance period, defined as a 5-step time window after the cue offset. Rank representations are obtained by averaging neural activity in this window for each rank of the cued item, collapsing across item identity. To reveal the underlying geometry, we perform dimensionality reduction on the rank representations. (A) Cumulative explained variance as a function of the number of principal components (PCs). (B,C) Classical multidimensional scaling (MDS) based on Euclidean distances, applied to human imaging data (adapted from [16]) and rank representations. Numbers 1–9 denote presentation-order ranks. (D) PCA visualization of rank representations in the top three PCs. The two shaded triangles represent an example of a local-rank plane and a global-rank plane. (E) Angles between rank planes. Within global denotes angles among global-rank planes; within local denotes angles among local-rank planes; global vs. local denotes angles between global- and local-rank planes. (F) Model fitting performance as a function of LGA and LGS, derived from a parametric representational geometry model. LGA: angle between local and global dimensions; LGS: unit length ratio between the two dimensions. The ideal orthogonal 2D geometry hypothesis assumes LGA=90°, LGS=1. See Appendix E.3 for details.

3.2 2D-like Neural Geometry Emerges in Hebb-RNNs

Human neural recordings reveal that sequence items are organized in an approximately 2D, orthogonal neural geometry, with two dimensions respectively encoding global (chunk-level) and local (within-chunk) rank [16]. This geometry enables the simultaneous representation of sequences across multiple

timescales. However, how such geometry emerges in a concrete model remains an open question; here, we investigate whether Hebb-RNNs exhibit a similar 2D neural geometry as observed experimentally.

we first analyze the neural geometry of Hebb-RNNs during the early post-cue delay period of the rank-recalling task (see Appendix E.1). Following Fan et al., we extract rank representations by averaging hidden-state activity across the early post-cue delay and all testing item-rank combinations, isolating rank-related structure while averaging out item-specific information. Using sequences with a 3-3-3 structure as an illustrative example, principal component analysis (PCA) reveals that rank representations are encoded in a low-dimensional representational space (Fig. 3A,D), and the top two PCs account for nearly 80% of the variance, substantially exceeding matched Gaussian null models ($p < 10^{-4}$; Appendix E.2). Following Fan et al., we further visualize the representational geometry using MDS (Fig. 3C), revealing two approximately orthogonal axes encoding global and local rank, closely resembling the structure observed in human neural data (Fig. 3B). This geometry also emerges robustly in Hebb-RNNs across different plasticity rules and with random stimuli (Appendix A.4.6).

To quantify this geometric organization (Fig. 3D), we define local-rank planes as those formed by rank representations within the same chunk (e.g., ranks 1, 2, and 3), yielding three local-rank planes. Analogously, global-rank planes are formed by rank representations aligned along the global axis (e.g., ranks 3, 6, and 9), resulting in three global-rank planes. We then compute plane angles: those between local- and global-rank planes are close to $90°$, whereas within-type angles were near zero, confirming an orthogonal coding scheme (Fig. 3E). Trial-label permutation controls confirmed this factorized organization, with within-factor planes significantly more parallel and between-factor planes more separated than under the null (all $p = 0.0002$; Appendix E.4). Additionally, we estimate LGA and LGS using Fan et al.'s parametric representational geometry model (Fig. 3F). Our Hebb-RNN yields LGA = $93°$, closely matching the ideal orthogonal prediction ($90°$) and broadly consistent with human data (LGA = $109°$). However, the model's LGS of 1.39 (>1) indicates a local precedence effect - local rank information is more discriminative than global - while humans (LGS = 0.9, <1) exhibit a global precedence effect [33, 16, 34] (See Sec. 4 for further discussion).

Collectively, these results demonstrate that a 2D neural geometry naturally emerges in Hebb-RNNs while performing rank recalling tasks, agreeing closely with experimental findings [16].

3.3 1D-to-2D Neural Geometric Folding in Hebb-RNNs

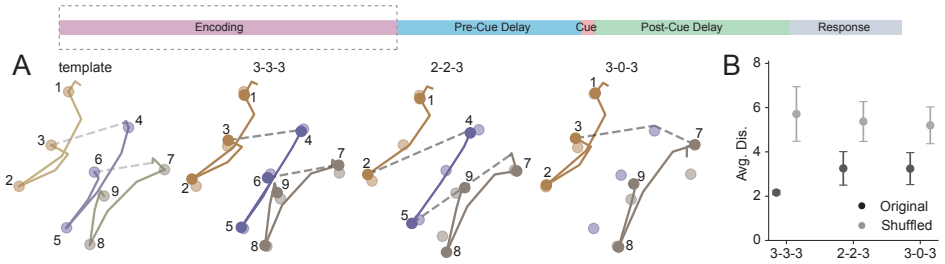

Figure 4: Rank Computation in Hebb-RNNs via Hierarchical Geometry. (A) From left to right, panels show the abstract rank geometry (template) identified during the encoding phase and example encoding trajectories for sequences with 3-3-3, 2-2-3, and 3-0-3 structures. The rank-geometry trajectory is obtained by averaging across diverse sequences with a 3-3-3 structure, with circles indicating rank positions. In the 3-0-3 configuration, "0" denotes a chunking signal presented without a subsequent stimulus input (see Appendix A.1). (B) Mean point-to-point Euclidean distance between each trial trajectory and the rank-geometry trajectory across different sequence structures. Shuffled controls are computed by randomly permuting time points within each block of the trial trajectory. Results are averaged over 10 seeds. See Appendix E.5.

How do abstract 2D neural geometry and fast Hebbian plasticity jointly enable the 1D-to-2D geometric folding observed in Hebb-RNNs, and how does this process give rise to hierarchical memory organization? To address these questions, we conduct a mechanistic analysis of computation in Hebb-RNNs.

Hierarchical geometry for rank computation. As temporal inputs do not carry explicit ordinal rank information, we examine how the Hebb-RNN integrates them to compute local and global

ranks. We first project population activity onto the 2D neural geometry identified during encoding (Appendix E.5). Neural states evolve in an orderly manner within a shared geometric manifold across different sequence structures (3-3-3, 2-2-3; Fig. 4A). Item presentation moves the state along a local rank dimension, while chunking signals reset the local position and advance the trajectory along a global rank axis. This interpretation is further supported by the 3–0–3 condition, in which a chunking signal occurs without a subsequent item, yet the global rank is correctly updated, showing that chunking signals alone can drive motion along the global dimension (intact trajectories plotted in Appendix E.6). Despite structural differences, neural trajectories across conditions consistently align with the averaged rank-geometry trajectory, significantly exceeding shuffled controls and confirming the reuse of a shared abstract scaffold (Fig. 4B). Together, these results suggest that 2D neural geometry integrates temporal inputs to perform abstract rank computation.

Memory storage in fast Hebbian weights. Secondly, we ask where item-rank conjunctive memories are stored: in neural hidden activities or in fast Hebbian weights. To address this question, we shuffle hidden activities immediately before cue onset, thereby eliminating any information carried by ongoing activity. As shown in Fig. 5A, this manipulation does not impair rank-generalization performance relative to control trials with intact activity. This result indicates that conjunctive memories are not maintained in neural activity. Consistent with this finding, directly probing the fast Hebbian weights $A \odot P$ with diverse stimuli just before cue onset reveals a clear 2D neural geometry (Fig. 5B). Together, these observations demonstrate that item-rank conjunctive memories in Hebb-RNNs are stored in fast Hebbian weights rather than in transient neural activity.

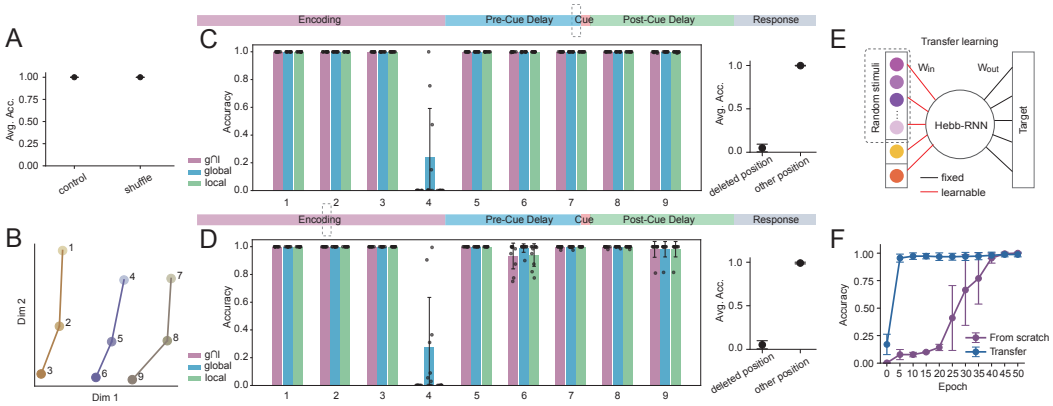

Figure 5: Memory Storage and Organization in Fast Plasticity Weights in Hebb-RNNs. (A) Control analysis of memory storage during the pre-cue delay. Memory is selectively disrupted in neural activity. In the shuffle condition, hidden activities in the Hebb-RNN are set to zero before cue onset, whereas in the control condition, hidden activities remain intact. (B) Hierarchical geometry embedded in fast Hebbian matrix. Rank representations are obtained immediately using corresponding stimulus representations to probe the Hebbian matrix before cue onset (see Appendix E.7 for details). (C) Performance of Hebb-RNNs under selective ablation of Hebbian memory during the pre-cue delay period. The left panel shows an example with ablation at the 4th-order memory component; the right panel shows averaged performance of ablated models versus intact models. Performance is averaged across all nine order positions and 10 random seeds. (D) Same as (C), but with memory ablation applied during the encoding period (see Appendix F for details). (E) Schematic of the transfer learning paradigm. Hebb-RNNs are initially trained on a rank-recalling task using one-hot stimuli, then transferred to a version of the task utilizing random Gaussian stimuli. During transfer, recurrent and readout weights remain frozen; only the input projections are learnable. (F) Learning dynamics. Comparison of task accuracy over training epochs between models employing transfer learning and those trained from scratch. Further details are provided in Appendix C.1.

Linear superposition of conjunctive memories. Finally, we examine how conjunctive memories are organized within the fast Hebbian matrix. During the pre-cue delay, we probe the Hebbian matrix with a stimulus representation $\mathbf{m}_4$ to recover the associated rank vector $\mathbf{n}_4$ (Appendix F). This item-rank association is then selectively removed by subtracting $\mathbf{n}_4\mathbf{m}_4^\top$ from the Hebbian matrix, yielding $A \odot (P - \mathbf{n}_4\mathbf{m}_4^\top)$. As shown in Fig. 5C, this manipulation selectively disrupts rank prediction for item 4 while leaving other items unaffected. Similar item-specific effects are observed

for other rank positions and whether ablation occurs during encoding or the delay period (Fig. 5C-D). These results suggest that conjunctive memories are encoded sequentially and independently, and are linearly superimposed in the fast Hebbian weight matrix.

1D-to-2D folding via Hebbian-geometric interaction. These results provide a clear picture of 1D-to-2D neural geometric folding in Hebb-RNNs. During encoding, stimulus and chunking signals drive the network state along a 2D neural geometry, enabling the computation of both global and local ranks. Simultaneously, fast Hebbian plasticity binds the computed ranks to incoming item representations, storing and maintaining them in the fast weight matrix. When cued, an item retrieves the stored rank information from the fast weights to generate accurate predictions. Thus, 1D-to-2D neural geometric folding in Hebb-RNNs is realized through the interaction between hierarchical neural geometry and fast Hebbian plasticity.

Transferability of the 1D-to-2D folding process. To investigate if the discovered folding mechanism serves as a reusable computational algorithm, we conduct a transfer experiment across distinct input regimes. We first pretrain the model on one-hot stimuli to establish the geometric scaffold, then freeze the recurrent and readout weights, retraining only the input projection on novel Gaussian random stimuli (Fig. 5E). The pretrained model adapts to these new statistics significantly faster than a control model trained from scratch (5 vs. 45 epochs, Fig. 5F). Crucially, visualizations confirm that rank computation in the transferred model continues to traverse the original 2D geometry (Appendix C.2). These results indicate that the 1D-to-2D folding process can function as an abstract algorithm capable of accelerating learning in novel environments.

3.4 Hebb-RNNs Learn 2D Geometry Without Predefined Chunking Signals

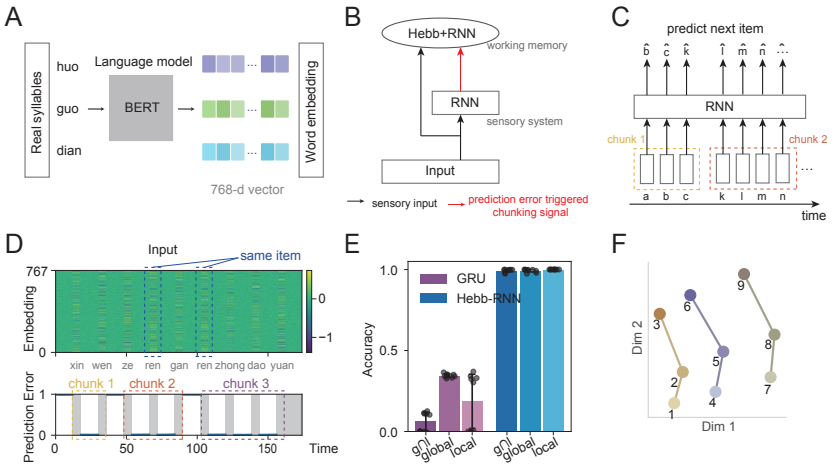

Figure 6: Learning 2D Neural Geometry in a Naturalistic Memory Recall Task without Explicit Chunking Signals. In the memory rank-recalling task, we use multisyllabic words as input, and no explicit chunking signals are provided. (A) Syllable embedding. A pretrained BERT language model is used to extract syllable embeddings (Appendix D.1). (B) Hybrid architecture. We combine a Hebb-RNN simulating working memory with a sensory prediction module. A vanilla RNN receives sensory inputs and generates prediction errors. When the prediction error exceeds a threshold (e.g., 0.8), a chunking signal is triggered and sent to the Hebb-RNN (which simulates working memory). (C) Unsupervised prediction. The vanilla RNN is trained via unsupervised next-item prediction, without explicit temporal structure information (Appendix G.1). At each time step, it receives an item (i.e., an embedding vector) and predicts the next item. The sequence has an internal temporal structure: syllables within a multisyllabic word are fixed, but the order of words is arbitrary. (D) Example trial and prediction error. The prediction error of a trained vanilla RNN on an example trial is shown. After each item is presented, the network outputs a probability distribution over the next item. The prediction error (blue lines in the lower panel) is computed as one minus the predicted probability assigned to the actual next item. (E)(F) Performance and geometry. Performance of GRU and Hebb-RNN on sequences containing chunks with fixed items (10 seeds), along with the two-dimensional hierarchical structure obtained from a Hebb-RNN using the method in Fig. 3C.

The tasks above use a predefined chunking signal in Hebb-RNNs, but how chunk structure can be inferred from the input and used to organize working memory remains unclear. In the brain, prediction errors, often arising in sensory or subcortical regions, may provide this signal. For example, in Fan et al.'s paradigm, the fixed syllable order within multisyllabic words allows the brain to exploit prior semantic knowledge to generate internal chunking signals. We adopt this naturalistic setting with real language inputs (Appendix D.1). Although syllable order per word is fixed, the same syllable appears in many words, making the task non-trivial. The network must infer global or local rank from context, not syllable identity alone - a demand for compositional generalization, not mere memorization.

Concretely, we first replaced the externally provided chunking signal with one derived from prediction error. We obtain context-independent syllable embeddings from a pretrained BERT model by embedding each syllable in isolation, (Fig. 6A) and introduce a sensory-prediction RNN to generate the chunking signal (Fig. 6B). This hybrid architecture preserves the Hebb-RNN's memory mechanism while making segmentation data-driven. The sensory-prediction RNN is trained in a self-supervised manner to predict the next syllable (Fig. 6C). Shown in Fig. 6D, after training, prediction error remains low within multisyllabic words - even when the same syllable (e.g., "ren") appears in different chunks, the model predicts correctly - but spikes at word boundaries. By thresholding the prediction error, we obtain a binary chunking signal for the Hebb-RNN.

Despite the naturalistic setting, our hybrid Hebb-RNN architecture achieves perfect performance, whereas a hybrid GRU architecture performs substantially worse (Fig. 6E). Moreover, the Hebb-RNN develops a 2D geometric scaffold (Fig. 6F) with LGA 94° and LGS 1.25 (Appendix D.2,Fig. S12). The same prediction-based segmentation remains effective with context-independent random representations, ruling out a dependence on semantic or contextual information in the embeddings (Appendix D.3).

Overall, our Hebb-RNN can organize real-word sequences into hierarchical memory representations without predefined chunking signals, as these signals can be naturally generated by a prediction module through self-supervised predictive learning.

4 Discussion

In this work, we use the Hebb-RNN - a recurrent neural network with fast Hebbian plasticity - to explain the hierarchical geometry of sequence memory. We show that: (1) Hebb-RNN achieves superior generalization in rank-recalling tasks, outperforming control models; (2) an abstract hierarchical geometry for ordinal information emerges, spanned by approximately orthogonal global and local rank axes that closely mirror human experimental findings; (3) the underlying computational mechanism is revealed, in which the geometry serves as a scaffold for abstract rank computation and fast plasticity binds and stores item–rank conjunctions, enabling 1D-to-2D neural geometrical folding of sequence memory; and (4) in naturalistic settings, Hebb-RNNs learn 2D neural geometry without externally provided chunking signals, demonstrating compositional generalization over multisyllabic word inputs. Together, our results provide a concrete, biologically plausible account of hierarchical sequence representation.

Related Works Fast Hebbian plasticity is widely proposed as a core neural mechanism supporting working and episodic memory [28, 29]. There is growing interest in using plastic neural networks to model high-level cognition. One approach introduces fast Hebbian plasticity into manually designed recurrent networks, for example, to explain capacity limits in working memory [30] or to reproduce contraction and repulsion effects in working memory tasks [35]. Another approach uses meta-learning to train neural networks to exploit synaptic plasticity or to discover effective plasticity rules for complex cognitive functions, including continual familiarity detection [36] and rapid knowledge reassembly [37–39]. Our work builds on this second line of research by meta-training a plastic recurrent network, but departs from prior studies by focusing on how temporal sequences are hierarchically represented and organized in memory.

Recent studies have highlighted the roles of network architecture, initialization, and optimization in shaping lazy and rich learning regimes in artificial and biological neural networks [40, 41]. Within the framework of Farrell et al. [40], our Hebb-RNN exhibits features consistent with rich learning, developing an approximately two-dimensional factorized geometry that captures the task's latent hierarchical structure. This geometry consistently emerges across different Hebbian update rules (Fig. S6) and several recurrent-weight initialization scales in successfully trained models, whereas

optimizer controls suggest some sensitivity to the training procedure (Sec. A.4.3). Unlike the relatively simple classification tasks commonly studied in the rich–lazy learning literature, hierarchical rank recall requires computing abstract ranks, binding them to arbitrary items, and flexibly retrieving items following a cue-operations that enable compositional generalization. Whether these computations necessarily depend on such a two-dimensional geometry, and how network architecture, initialization, and optimization shape its emergence, remain open questions.

Limitation and Future Work Several limitations of the current study point to directions for future research. (1) While recent evidence suggests that 2D geometry can emerge in the brain without explicit hierarchical prediction [42], its underlying origin remains elusive. It is unclear whether these representations spontaneously self-organize under memory capacity constraints or instead reflect the reuse of pre-existing structural scaffolds shaped by hierarchical inputs, such as language. Hebb-RNNs offer a promising framework for probing this open question. (2) The scalability of our findings to more complex hierarchies (e.g., more than two levels) remains to be determined; as a first step, we have obtained preliminary results in three-level hierarchies (see Appendix B.2). (3) Bridging the gap between our BPTT-based meta-learning framework and fully biologically plausible plasticity rules remains a key challenge for improving biological plausibility. (4) Here, we compare only 1D and 2D neural geometries. Alternative coding schemes, such as sparse or simplex-like representations, may also explain the experimental findings and support compositional generalization. Future work could evaluate these alternatives against experimental observations. (5) The learned representational geometry may depend on optimization and training choices [40]. A systematic characterization of optimizer, regularization, and other training regimes remains for future work.

Additionally, humans typically exhibit a global precedence effect across visual [33, 43, 44], auditory [34], and working memory domains [16], whereas our model exhibits a local precedence effect. Although multisyllabic word inputs reduce the LGS from 1.39 to 1.25, a discrepancy with human data (LGS = 0.9) remains. Control analyses reveal that this global–local bias is sensitive to task-related signals: stronger chunk-boundary inputs enhance global separation, whereas stronger local-rank targets enhance local separation (Appendix H). These findings suggest that the relative weighting of global and local information is not fixed, but depends on the signals guiding representation learning. In the brain, such weighting may be dynamically regulated by endogenous top-down modulation [43, 45], potentially contributing to the global precedence observed in humans. Incorporating top-down control into hierarchical memory models may therefore help explain how global–local precedence emerges and is regulated, beyond the formation of the underlying two-dimensional geometry.

Finally, our results suggest a potential way to incorporate dynamic, multiscale positional structure into Transformers. Future work could leverage prediction errors to trigger chunking signals that dynamically segment sequences into structured units. Integrating this mechanism with fast weights in Transformers may enable more flexible, brain-like memory organization and sequence processing across multiple scales [46, 47].

Acknowledgments and Disclosure of Funding

This work was supported by the National Science and Technology Innovation 2030 Major Program (No. 2021ZD0203705, Y.Y. Mi), the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM504, H. Luo), the National Natural Science Foundation of China (No. 62336007 / 32541018, B. Hong), the National Natural Science Foundation of China (No. 32541013, H. Luo), and the National Science and Technology Innovation 2030 Major Project (No. 2021ZD0204103, H. Luo).

References

[1] Daniel L. Schacter, Kenneth A. Norman, and Wilma Koutstaal. The cognitive neuroscience of constructive memory. *Annual review of psychology*, 49:289–318, 1998.

[2] Mirko Thalmann, Alessandra S. Souza, and Klaus Oberauer. How does chunking help working memory? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 45:37–55, 2019.

[3] Nai Ding. Sequence chunking through neural encoding of ordinal positions. *Trends in Cognitive Sciences*, 29:641–654, 2025.

[4] Christopher Baldassano, Janice Chen, Asieh Zadbood, Jonathan W. Pillow, Uri Hasson, and Kenneth A. Norman. Discovering Event Structure in Continuous Narrative Perception and Memory. *Neuron*, 95(3):709–721.e5, August 2017.

[5] Gabriel A Radvansky and Jeffrey M Zacks. Event boundaries in memory and cognition. *Current Opinion in Behavioral Sciences*, 17:133–140, October 2017.

[6] Jeffrey M. Zacks. Event Perception and Memory. *Annual Review of Psychology*, 71(Volume 71, 2020):165–191, January 2020.

[7] Jeffrey M. Zacks, Nicole K. Speer, Khena M. Swallow, Todd S. Braver, and Jeremy R. Reynolds. Event perception: A mind-brain perspective. *Psychological Bulletin*, 133(2):273–293, March 2007.

[8] Christopher A. Kurby and Jeffrey M. Zacks. Segmentation in the perception and memory of events. *Trends in Cognitive Sciences*, 12(2):72–79, February 2008.

[9] Alexander J. Barnett, Mitchell A. Nguyen, James Spargo, Reesha Yadav, Brendan I. Cohn-Sheehy, and Charan Ranganath. Hippocampal-cortical interactions during event boundaries support retention of complex narrative events. *Neuron*, 112:319–330.e7, 2023.

[10] Momchil S. Tomov, Samyukta Yagati, Agni Kumar, Wanqian Yang, and Samuel J. Gershman. Discovery of hierarchical representations for efficient planning. *PLoS Computational Biology*, 16(4):e1007594, 2020.

[11] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97, 1956.

[12] Nelson Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *The Behavioral and Brain Sciences*, 24(1):87–114, 2001.

[13] Nelson Cowan. Working-memory capacity limits in a theoretical context. In Chizuko Izawa and Nobuo Ohta, editors, *Human Learning and Memory: Advances in Theory and Application: The 4th Tsukuba International Conference on Memory*, pages 155–175. Lawrence Erlbaum Associates Publishers, 2005.

[14] Dennis Norris and Kristjan Kalm. Chunking and data compression in verbal short-term memory. *Cognition*, 208:104534, 2021.

[15] Aneri Soni and Michael J Frank. Adaptive chunking improves effective working memory capacity in a prefrontal cortex and basal ganglia circuit. *eLife*, 13:RP97894, 2025.

[16] 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*, 9:360–375, 2025.

[17] Honey L. H. Ng and Murray T. Maybery. Grouping in short-term verbal memory: Is position coded temporally? *Quarterly Journal of Experimental Psychology*, 55:391 – 424, 2002.

[18] 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*, 88:2–19, 2015.

[19] Albert Tsao, Jørgen Sugar, Li Lu, Cheng Wang, James J. Knierim, May-Britt Moser, and Edvard I. Moser. Integrating time from experience in the lateral entorhinal cortex. *Nature*, 561(7721):57–62, 2018.

[20] Benjamin R. Kanter, Christine M. Lykken, Ignacio Polti, May-Britt Moser, and Edvard I. Moser. Event structure sculpts neural population dynamics in the lateral entorhinal cortex. *Science*, 388 6754:eadr0927, 2025.

[21] Mike Page and Dennis Norris. The primacy model: a new model of immediate serial recall. *Psychological review*, 105 4:761–81, 1998.

[22] Neil Burgess and Graham J. Hitch. Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review*, 106(3):551–581, 1999.

[23] Richard N. A. Henson. Short-term memory for serial order: The start-end model. *Cognitive Psychology*, 36:73–137, 1998.

[24] Sarthak Chandra, Sugandha Sharma, Rishidev Chaudhuri, and Ila Rani Fiete. Episodic and associative memory from spatial scaffolds in the hippocampus. *Nature*, 638:739 – 751, 2025.

[25] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

[26] James C. R. Whittington, William Dorrell, Tim 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*, 113:321–333.e6, 2024.

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

[28] Mark G Stokes. 'activity-silent'working memory in prefrontal cortex: a dynamic coding framework. *Trends in cognitive sciences*, 19(7):394–405, 2015.

[29] Anders Lansner, Florian Fiebig, and Pawel Herman. Fast hebbian plasticity and working memory. *Current Opinion in Neurobiology*, 83:102809, 2023.

[30] Florian Fiebig, Pawel Herman, and Anders Lansner. An indexing theory for working memory based on fast hebbian plasticity. *eNeuro*, 7(2):ENEURO.0374–19.2020, 2020.

[31] David Clewett, Camille Gasser, and Lila Davachi. Pupil-linked arousal signals track the temporal organization of events in memory. *Nature Communications*, 11(1):4007, August 2020.

[32] Yi Pu, Xiang-Zhen Kong, Charan Ranganath, and Lucia Melloni. Event boundaries shape temporal organization of memory by resetting temporal context. *Nature Communications*, 13(1):622, February 2022.

[33] David Navon. Forest before trees: The precedence of global features in visual perception. *Cognitive psychology*, 9(3):353–383, 1977.

[34] Lisa D Sanders and David Poeppel. Local and global auditory processing: Behavioral and erp evidence. *Neuropsychologia*, 45(6):1172–1186, 2007.

[35] Francesca Schönsberg, Davide Giana, Yukti Chopra, Mathew E Diamond, and Sebastian Goldt. Diverse perceptual biases emerge from hebbian plasticity in a recurrent neural network model. *Neuron*, 113(21):3673–3684, 2025.

[36] Danil Tyulmankov, Guangyu Robert Yang, and LF Abbott. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. *Neuron*, 110(3):544–557, 2022.

[37] Thomas Miconi. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife*, 6:e20899, 2017.

[38] Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: Training plastic neural networks with backpropagation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3559–3568. PMLR, 2018.

[39] Thomas Miconi and Kenneth Kay. Neural mechanisms of relational learning and fast knowledge reassembly in plastic neural networks. *Nature Neuroscience*, 28(2):406–414, 2025.

[40] Matthew Farrell, Stefano Recanatesi, and Eric Shea-Brown. From lazy to rich to exclusive task representations in neural networks and neural codes. *Current Opinion in Neurobiology*, 83:102780, 2023.

[41] Timo Flesch, Keno Juechems, Tsvetomira Dumbalska, Andrew M. Saxe, and Christopher Summerfield. Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron*, 110:1258 – 1270.e11, 2022.

[42] Xiangjuan Ren, Muzhi Wang, Tingting Qin, Fang Fang, Aming Li, and Huan Luo. "compressive learning" scaffolds higher-order network structure to enhance human knowledge acquisition. *bioRxiv*, 2025.

[43] Moshe Bar, Karim S Kassam, Avniel Singh Ghuman, Jasmine Boshyan, Annette M Schmid, Anders M Dale, Matti S Hämäläinen, Ksenija Marinkovic, Daniel L Schacter, Bruce R Rosen, et al. Top-down facilitation of visual recognition. *Proceedings of the national academy of sciences*, 103(2):449–454, 2006.

[44] Xiaolong Zou, Zilong Ji, Tianqiu Zhang, Tiejun Huang, and Si Wu. Visual information processing through the interplay between fine and coarse signal pathways. *Neural Networks*, 166:692–703, 2023.

[45] Timothy F Brady and George A Alvarez. Hierarchical encoding in visual working memory: Ensemble statistics bias memory for individual items. *Psychological science*, 22(3):384–392, 2011.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[47] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.

[48] Barak Blumenfeld, Son Preminger, Dov Sagi, and Misha Tsodyks. Dynamics of memory representations in networks with novelty-facilitated synaptic plasticity. *Neuron*, 52:383–394, 2006.

A Experimental Protocol and Model Configuration for Rank Recalling Task

A.1 Rank Recalling Task Settings

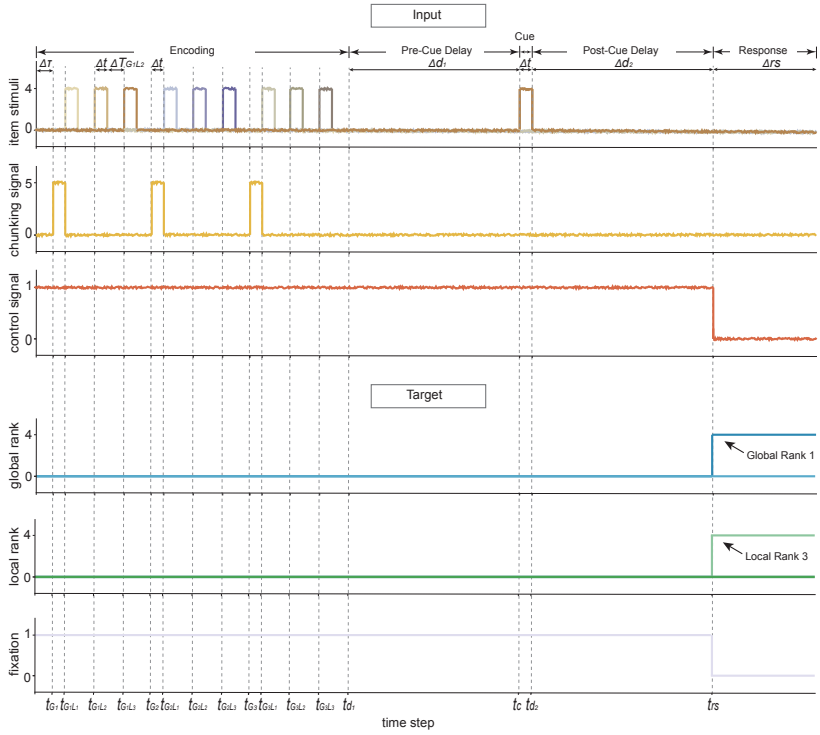

Figure S1: An Example of the Input and Target Sequences in the Rank-Recall Working Memory Task. In this example, a 3–3–3 sequence is presented. The upper three panels depict the input channels, corresponding to the item stimuli, the chunking signal, and the control signal, respectively. The lower panels show the output targets, including the global-rank target, local-rank target, and fixation target. When the third cue is presented, the global rank is 1, whereas the local rank is 3.

Our objective is not merely to train networks that succeed on the rank-recall working memory task, but to uncover the neural mechanisms underlying hierarchical geometric representations in sequence memory. To this end, we intentionally simplify the task structure as much as possible—without sacrificing generality—while preserving the core features of the tasks used in human experiments [16].

As shown in Fig. S1, the rank-recalling task consists of five successive phases: an encoding period, a pre-cue delay, a cue presentation, a post-cue delay, and a response period. During encoding, sequential input syllables are represented as one-hot vectors of dimension 20. Each syllable is presented for a stimulus intensity factor of 4 and a duration of $\Delta t = 5$, with inter-syllable intervals randomly sampled from a uniform distribution $\Delta T \sim [5, 12]$.

Two additional scalar input signals are provided: a chunking signal and a control signal. The chunking signal marks the beginning of each chunk and is presented immediately before the first syllable of the chunk. It takes a value of 5 when active and 0 otherwise, with a fixed duration of $\Delta t = 5$ (except in Fig. 2G, where the network is trained and tested with the chunking signal set to zero throughout the entire trial). Sequence-structure configuration of 2-3-4, 2-4-3, 3-2-4, 3-4-2, 4-2-3, 4-3-2 and 3-3-3 (as shown in Fig. 2C) are applied during training. The first chunking signal appears after a random interval from the beginning of the trial, randomly sampled from a uniform distribution $\Delta\tau \sim [5, 7]$ to allow the network to be active. The control signal indicates whether the network should withhold or produce a response, mimicking the fixation cue used in behavioral experiments. Specifically, the control signal is set to 1 during all non-response periods and switches to 0 at the

onset of the response period. All input channels are corrupted with additive random noise sampled from a Gaussian distribution with mean equaling to 0 and standard deviation equaling to 0.01.

Following the encoding phase, a pre-cue delay is introduced, with its duration randomly sampled from $\Delta d_1 \sim [85, 95]$. After this delay, a cue item is randomly selected from the syllables presented in the current trial and is shown for $\Delta t = 5$. The cue is immediately followed by a post-cue delay, whose duration is sampled from $\Delta d_2 \sim [80, 90]$. Once the control signal switches to 0, the response period begins, with a random duration of $\Delta r_s \sim [25, 30]$.

Both global and local rank targets are encoded using one-hot vectors. Before the response period, all target units are set to zero. During the response, the units corresponding to the correct global and local ranks are set to 4. In addition, a fixation cue is included as a scalar input: it is set to 1 before the response and switches to 0 when the response begins, facilitating learning by explicitly signaling the appropriate time for response generation.

In Fig. 4, we introduce a 3-0-3 sequence-structure configuration, in which "0" denotes a chunking signal presented without subsequent stimulus input. Here we show an input-target example applying this configuration (Fig. S2).

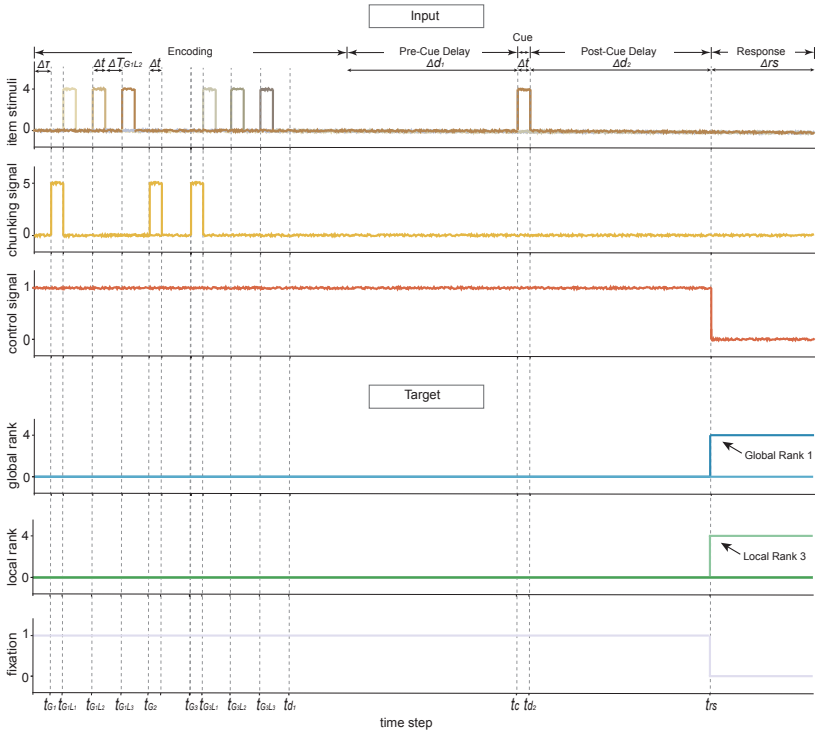

Figure S2: Input and Target for a 3-0-3 Sequence in the Rank-Recalling Task. In this configuration, a chunking signal delivered without stimulus items. When the third cue appears, the global rank is 1, but the local rank is 3.

A.2 Model and Training Details for the Rank Recalling Task

Our Hebb-RNN consists of a single recurrent hidden layer with 300 units. The time constant τ governing hidden-state dynamics is set to 2. The fast Hebbian learning rate η is 0.1; in practice, the Hebb-RNN exhibits robust performance across a broad range of η values. The recurrent weights W_{rec}, input weights W_{in}, and plasticity coefficient matrix A are randomly initialized using Xavier initialization, while the fast plasticity matrix P is initialized to zero at the beginning of each trial. All bias terms are initialized to zero.

Control models, including vanilla RNNs, GRUs, LSTMs and variant RNNs with short-term plasticity (STP-RNNs) [27], employ 300 hidden units. All recurrent models use Xavier initialization and

share the same inputs and outputs as the Hebb-RNN. The Transformer model is configured with an embedding dimension of 256, a feedforward network (FFN) hidden size of 1024, 3 decoding layers, sinusoidal positional encoding, 4 attention heads, a dropout rate of 0.1, and a maximum sequence length of 500.

The STP-RNN incorporates short-term synaptic plasticity dynamics, in which the membrane potential evolves as:

$$\tau \frac{d\mathbf{h}(t)}{dt} = -\mathbf{h}(t) + (W_{rec} \cdot \mathrm{diag}(\mathbf{x}(t) \odot \mathbf{u}(t)))\mathbf{r}(t) + W_{in} I(t) + \mathbf{b}, \tag{4}$$

where $\mathbf{x}(t)$ and $\mathbf{u}(t)$ denote the synaptic resource availability and utilization, respectively, governed by:

$$\begin{aligned} \frac{d\mathbf{x}(t)}{dt} &= \frac{1 - \mathbf{x}(t)}{\tau_x} - (\mathbf{x}(t) \odot \mathbf{u}(t)) \odot \mathbf{r}(t), \\ \frac{d\mathbf{u}(t)}{dt} &= \frac{U - \mathbf{u}(t)}{\tau_u} + U(1 - \mathbf{u}(t)) \odot \mathbf{r}(t). \end{aligned} \tag{5}$$

To model short-term facilitation (STF)-dominant dynamics which are widely observed in prefrontal cortex, the time constants are set to $\tau_x = 120$ and $\tau_u = 360$, with baseline utilization $U = 0.15$. These parameters yield $\tau_u > \tau_x$, a regime characteristic of facilitating synapses as Mongillo et al. (2008) proposed.

All models are trained to minimize a regression loss for the rank-recalling task using backpropagation through time, implemented in PyTorch. Training is conducted using the Adam optimizer with a learning rate of 0.001, default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$), a dataset of 3000 samples, 50 epochs, and a batch size of 32. Each type of model is trained and evaluated across 10 random seeds. The regularization strength α in the loss function is set to 2.0 during the pre-cue and post-cue delay periods, and to 0 otherwise.

A.3 Testing Details for the Rank Recalling Task

When testing, we fix the beginning interval $\Delta\tau$ to 6, all the inter-syllable intervals ΔT to 7, the pre-cue delay period Δd_1 to 90, the post-cue delay period Δd_2 to 85, and the response period Δr_s to 27 for all the trials. Additive noise is sampled from the same Gaussian distribution during training. The Hebbian matrix and all the hidden layer activities are set to zero at the beginning of every trial.

In Fig. 2C, E-G, Fig. 5A, C-D, F, Fig. 6E, Fig. S3A-B, Fig. S6A-B, Fig. S7B and Fig. S8, we report test accuracies for different model types and configurations. For each random seed, we construct non-overlapping training and testing datasets such that item–rank combinations in the test set are never seen during training. Accuracies are computed using the predicted local and global ranks at the final time step of the response period, and the mean and standard deviation are reported across 10 seeds for each type of model.

Model performance is evaluated using three metrics: joint accuracy (both local and global ranks correct), global-rank accuracy, and local-rank accuracy. In Fig. 2C, G, Fig. S3A–B, Fig. S6A–B and Fig. S7B we randomly select a sequence-structure configuration from the training set (see Fig. 2D) and randomly choose the cued item for each of 200 trials. In Fig. 2E–F, we evaluate generalization by testing on sequence-structure configurations (2–2, 3–3, 2–2–3, 2–2–4, 3–3–4, and 4–4–4; see Fig. 2D) that are not used during training, again with randomly selected cued items.

In Fig. 5C–D, Fig. S7B, and Fig. S8, we focus on the 3–3–3 sequence structure and group trials into nine conditions defined by the cued item's sequential position (3 global ranks × 3 local ranks). For each group, 10 trials are sampled to compute group-specific accuracies. Furthermore, in Fig. 5C-D, for ablation on each position, we record the accuracy of the perturbed position alongside the mean accuracy across all other non-perturbed positions. These metrics are subsequently averaged across all ablation cases for each network seed. And in Fig. 5A, we average the accuracies of each trial group (that is, averaging the nine accuracies in Fig. S7), and compare this mean accuracy of the named shuffled condition with control condition, in which no ablation is applied.

A.4 Additional Results in Rank Recalling Tasks

A.4.1 Failure of Control Models Under Increased Capacity and Extended Training

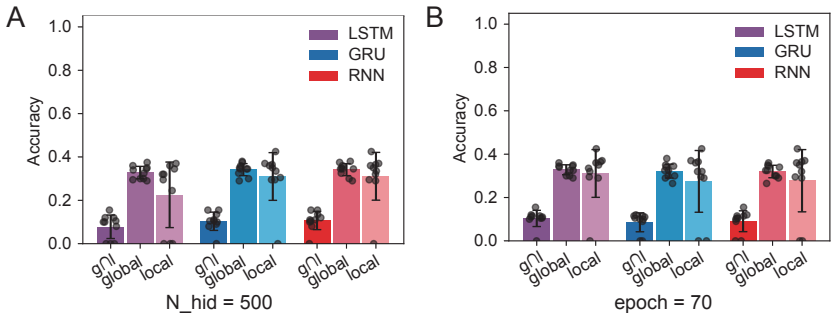

Figure S3: Performance of Control Models with Substantially Larger Capacities and More Training Epochs. (A) The hidden layer size is increased to 500 units for every model. (B) The number of training epochs is increased to 70 for all models.

Building upon the results presented in Fig. 2C, we further demonstrate that LSTM, GRU, and vanilla RNN models fail to reliably predict either rank, even when trained with substantially larger capacities and more training epochs in Fig. S3.

A.4.2 Sensitivity Analysis of the Hebbian Learning Rate and Network Size

We conduct an additional analysis to examine the sensitivity of Hebb-RNN performance to the fast Hebbian learning rate η. The results reveal that the model achieves perfect test accuracy at $\eta = 0.05$ and $\eta = 0.10$, whereas performance degrades at both smaller and larger values of η. At $\eta = 0.15$, only a subset of networks successfully learn the task, yielding a mean accuracy of approximately 53.3%. These failure modes underscore the critical balance required between recurrent activity and fast Hebbian memory. The test accuracy and representational geometry across all evaluated η values are summarized in Table S1.

Table S1: Test Accuracy and Representational Geometry of Hebb-RNNs across Different Hebbian Learning Rates η. Accuracy is reported as mean $\pm$ std across seeds.

η	Test Accuracy (%)
0.01	7.1 ± 5.1
0.05	100.0 ± 0.0
0.10	100.0 ± 0.0
0.15	53.3 ± 46.8
0.50	10.0 ± 30.0

In addition, we examined the sensitivity of Hebb-RNN performance to the number of hidden units while keeping all other settings identical to those in Fig. 2C. Test accuracy was $24.3\% \pm 17.5\%$ with 100 hidden units, but reached $100.0\% \pm 0.0\%$ with 200, 300, and 500 hidden units.

A.4.3 Effects of optimizer choice and recurrent weight initialization gain on neural representations

We first examined whether the low-dimensional neural representations observed in Hebb-RNNs depend on the choice of optimizer. Replacing Adam with standard SGD without momentum under the original training setup generally failed to converge across learning rates from 10^{-4} to 10^{-1}. To improve convergence, we introduced gradient clipping (0.5), momentum (0.9), extended training to 300 epochs, and a stepwise learning-rate schedule (10^{-2}, 3×10^{-3}, 9×10^{-4}, and 2.7×10^{-4}). The tuned SGD procedure achieved a maximum validation accuracy of 93.75% and a test accuracy of $90.10\% \pm 11.44\%$ across five seeds (range: 70.50%–99.50%; Table S2).

Table S2: Task performance and variance concentration across optimizer and recurrent-initialization conditions. Hidden activity was evaluated on a controlled 3×3 global–local rank scaffold with 20 trials per condition. Accuracy and explained variance are reported as mean $\pm$ SD across converged runs. PC1–2 and PC1–3 denote the cumulative variance explained by the first two and three principal components, respectively. Converged runs indicate the number of successfully converged training runs relative to the total number of runs (e.g., 2/5 indicates two converged runs out of five). All initialization-gain conditions used Adam.

Condition	Converged runs	Accuracy	PC1–2 variance	PC1–3 variance
Adam, $g = 1.0$ (main-text)	$10/10$	$100.00\% \pm 0.00\%$	$71.22\% \pm 13.69\%$	$88.20\% \pm 6.03\%$
Tuned SGD	$5/5$	$90.10\% \pm 11.44\%$	$70.07\% \pm 5.28\%$	$79.49\% \pm 3.59\%$
Adam, $g = 0.5$	$4/5$	$100.00\% \pm 0.00\%$	$62.31\% \pm 2.23\%$	$84.94\% \pm 1.73\%$
Adam, $g = 0.7$	$5/5$	$100.00\% \pm 0.00\%$	$67.29\% \pm 10.98\%$	$86.33\% \pm 5.30\%$
Adam, $g = 1.2$	$5/5$	$100.00\% \pm 0.00\%$	$69.25\% \pm 10.72\%$	$88.41\% \pm 4.67\%$
Adam, $g = 1.5$	$2/5$	$100.00\% \pm 0.00\%$	$82.61\% \pm 16.40\%^*$	$91.81\% \pm 7.47\%$

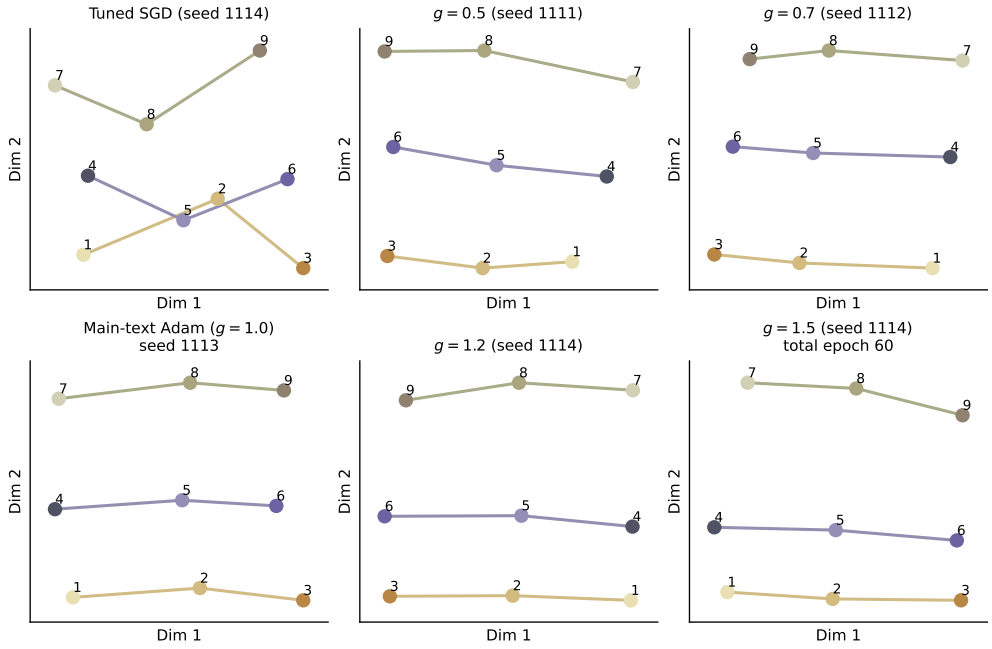

Figure S4: Rank representations under the tuned SGD procedure and recurrent-initialization gain controls. Visualization follows the same procedure as in Fig. 3C. Each panel was independently rotated to display its geometry clearly.

Applying the same PCA analysis as in Fig. 3C, the first two and three PCs explained $70.07\% \pm 5.28\%$ and $79.49\% \pm 3.59\%$ of the variance, respectively (Table S2). The PC1–2 variance was comparable to that of Adam-trained models. However, the projected representations did not consistently exhibit the orderly, approximately orthogonal 2D global–local rank geometry observed with Adam (Fig. S4A). This difference may partly reflect the lower task performance of SGD-trained models rather than an intrinsic effect of the optimizer.

We further examined the effect of recurrent-weight initialization gain by varying g from the main-text setting ($g = 1.0$) to 0.5, 0.7, 1.2, and 1.5, using Adam with otherwise matched training settings. Some runs exhibited training instability, characterized by persistent oscillations in hidden activity and global/local outputs, accompanied by near-chance task performance. All converged runs with $g = 0.5$, 0.7, or 1.2 achieved 100% test accuracy. For $g = 1.5$, the two remaining runs initially reached 39.75% and 88.50% after 50 epochs, but both achieved 100% after 8–11 additional epochs with a reduced learning rate (3×10^{-4}) and a reset Adam optimizer.

Across converged models, the leading PCs captured substantial variance, with variance concentration generally increasing at larger gains (Table S2). Representative projections retained an orderly, approximately 2D global–local rank geometry across gain conditions, despite variations in spacing and orientation (Fig. S4). Interestingly, the increasing variance concentration with gain does not straightforwardly align with previous accounts associating larger initialization scales with higher-dimensional representations under lazy learning [40]. This apparent discrepancy may reflect differences in task demands, recurrent architecture, or dimensionality measures, warranting further investigation.

A.4.4 Extended Analysis for Generalization on Longer and Deeper Sequences

We conduct additional experiments to evaluate the model on longer and deeper sequences (Table S3, Table S4). The results show that the network fails to predict the global rank in the fourth chunk, where no corresponding readouts are available. Importantly, local predictions remain accurate. Similar findings are observed in Generalization to Longer Chunks (5-2-2, 2-5-2, 2-2-5), highlighting the model's ability to generalize to deeper and longer sequences.

Table S3: Generalization to Longer Chunks. Test accuracies are averaged over sequences containing a chunk of length 5 (5-2-2, 2-5-2, 2-2-5), across 4 seeds.

Cued Item	**Global Accuracy (%)**	**Local Accuracy (%)**
Local rank ≤ 4	100.0 ± 0.0	100.0 ± 0.0
Local rank = 5	100.0 ± 0.0	0.0 ± 0.0

Table S4: Generalization to Deeper Chunks. Test accuracies are averaged over sequences with 4 chunks (2-2-2-3, 2-2-3-2, 2-3-2-2, 3-2-2-2), across 4 seeds.

Cued Item	**Global Accuracy (%)**	**Local Accuracy (%)**
In the first 3 chunks ≤ 4	100.0 ± 0.0	100.0 ± 0.0
In the 4th chunk	0.0 ± 0.0	100.0 ± 0.0

We further evaluated substantially longer sequences using 4-4-4-4 and 4-4-4-4-4 configurations. Because global ranks beyond the third chunk fall outside the trained readout range, accuracy was computed only over the first 12 items. Under this evaluation, accuracy decreased from 92.6% for 4-4-4 to 76.4% for 4-4-4-4 and 34.3% for 4-4-4-4-4.

A.4.5 Bootstrap Confidence Intervals for Best-Fitting LGA and LGS

Building upon the best-fitting parameters identified in Fig. 3F, we further assess the robustness of the estimated LGA and LGS via bootstrap resampling ($N = 100$). With the best-fitting LGS fixed, networks are resampled with replacement to re-estimate LGA across bootstrap iterations; the procedure is then repeated with the best-fitting LGA fixed to re-estimate LGS (more details in Appendix E.3). The resulting 95% confidence intervals are narrow and do not overlap with the parameter values corresponding to the 1-D geometry (LGA = 0°), but overlap with 2-D orthogonal (LGA = 90°) hypotheses, confirming that the 2-D hierarchical geometry is robustly estimated and not an artifact of sampling variability (Fig. S5).

A.4.6 Robustness of Hebb-RNNs Across Stimulus Statistics and Hebbian Learning Rules

As shown in Fig. S6, we further evaluate Hebb-RNNs using stimuli randomly sampled from a Gaussian distribution and a range of Hebbian plasticity update rules. The results demonstrate that Hebb-RNNs remain effective under randomly sampled stimulus conditions (Fig. S6A). However, for certain random seeds, test accuracy reaches only approximately 70%. This performance degradation is likely due to correlations among randomly sampled stimuli, which can interfere with memory storage and retrieval in fast Hebbian memory. Future work may address this limitation by incorporating novelty-based Hebbian learning rules [48].

To further quantify the effect of stimulus similarity, we systematically varied pairwise item similarity by generating each stimulus as $\mathbf{x}_i = \sqrt{\rho}\mathbf{z}_0 + \sqrt{1-\rho}\mathbf{z}_i$, where $\mathbf{z}_0$ is shared across items and $\mathbf{z}_i$ is

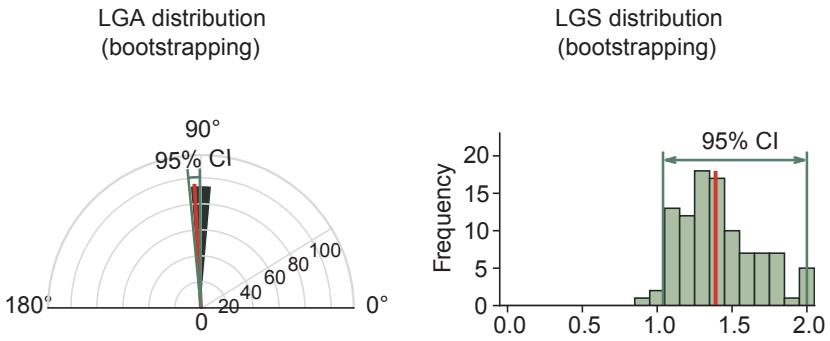

Figure S5: Bootstrap Distributions of LGA (LGS Fixed at 1.39, Which Is the Best-Fitting LGS in Fig. 3F) and LGS (LGA Fixed at 93°, Which Is the Best-Fitting LGA in Fig. 3F), Each Based on 100 Bootstraps.

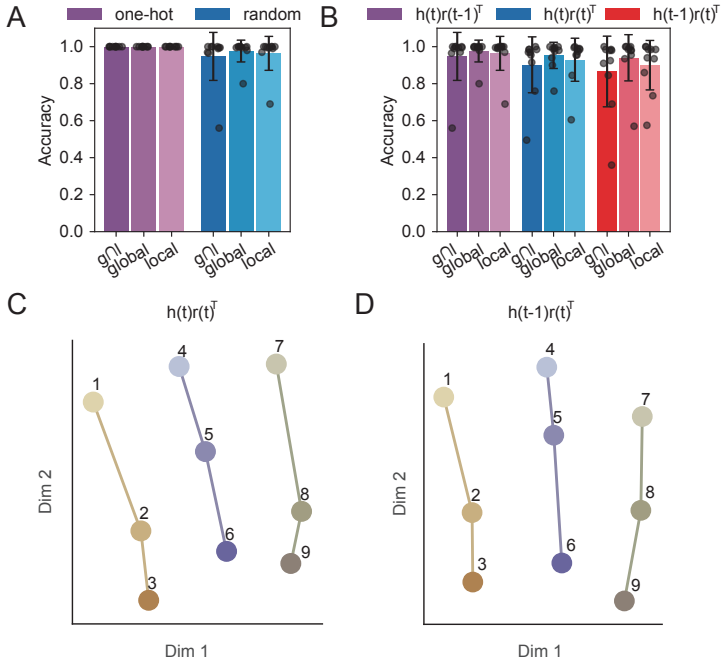

Figure S6: Control Studies of Stimuli and Hebbian Plasticity Rules in Hebb-RNNs. (A) Generalization performance of Hebb-RNNs with stimuli randomly sampled from a Gaussian distribution. Gaussian distribution has a mean of 0 and a standard deviation of 1. (B) Generalization of Hebb-RNNs under different Hebbian plasticity rules. From left to right: Hebbian rules with $\Delta P(t) = \mathbf{h}(t)\mathbf{r}(t-1)^T$, $\Delta P(t) = \mathbf{h}(t)\mathbf{r}(t)^T$, and $\Delta P(t) = \mathbf{h}(t-1)\mathbf{r}(t)^T$. (C) 2D hierarchical geometry in Hebb-RNNs with the Hebbian plasticity rule $\Delta P(t) = \mathbf{h}(t)\mathbf{r}(t)^T$. (D) 2D hierarchical geometry in Hebb-RNNs with the Hebbian plasticity rule $\Delta P(t) = \mathbf{h}(t-1)\mathbf{r}(t)^T$. Data visualization in (C) and (D) follows [16].

item-specific. Test accuracy remained near ceiling for $\rho \leq 0.7$ ($\geq 99.4\%$), but dropped sharply to $11.4\% \pm 10.4\%$ at $\rho = 0.9$. Thus, performance is robust across a broad range of stimulus similarities but degrades for highly correlated inputs.

Beyond stimulus statistics, we also examine the robustness of Hebb-RNNs to variations in the Hebbian update rule under randomly sampled inputs. For example, when the fast Hebbian matrix

$P(t)$ is updated online using the rule

$$P(t+1) = P(t) + \eta \mathbf{h}(t)\mathbf{r}(t)^T, \tag{6}$$

the model continues to generalize well (Fig. S6B). Moreover, as shown in Fig. S6C–D, hierarchical geometric representations consistently emerge in Hebb-RNNs across different Hebbian plasticity rules (more details in Appendix E.1).

Taken together, these results indicate that the emergence of 2D hierarchical rank geometry is not tied to specific stimulus distributions or a particular fast plasticity formulation in some degree. Instead, it reflects a more general computational property of Hebb-RNNs that is robust across input types and Hebbian learning rules.

A.4.7 Role of Slow Recurrent Weights in Rank Computation During Encoding

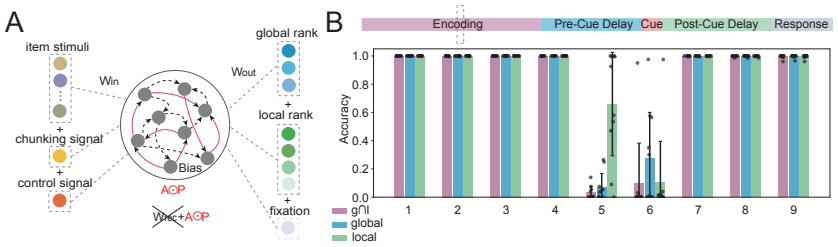

Figure S7: Performance of Hebb-RNNs under Selective Ablation of Slow Recurrent Connections during the Encoding Period. (A) Schematic of selective ablation in Hebb-RNNs. (B) performance under an example ablation condition. During the encoding period when item 5 is presented, the slow recurrent connections are selectively reset to zero, and are subsequently restored when item 6 is presented. Accuracies are averaged across 10 seeds.

Fig. S7 shows a selective ablation study on the slow recurrent weights in Hebb-RNNs. Specifically, the slow recurrent connections are transiently removed during the encoding phase of item 5 and restored at the subsequent item, without resetting Hebbian matrix or hidden layer activities to zero. We examine the resulting generalization performance under this condition, and compare it with generalization performance when ablations are applied at other encoding positions. These results provide additional insights into how the hierarchical geometry involves rank computation.

A.4.8 Activity-Silent Information Storage in the Hebbian Matrix During Maintenance

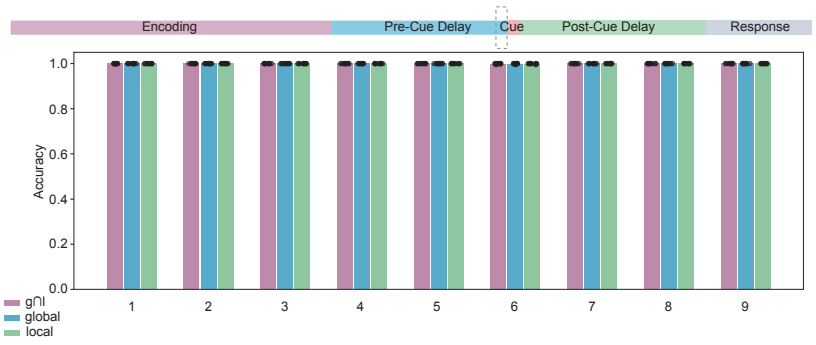

Figure S8: Performance of Hebb-RNNs under Ablation of Hidden Layer Activities prior to Cue Onset. Immediately before cue onset, the hidden layer activities are selectively reset to zero. Generalization performance under this ablation condition is shown, with accuracies averaged across 10 random seeds.

In addition to Fig. 5A, Fig. S8 shows an example ablation study on the hidden layer activities prior to cue onset in Hebb-RNNs. Specifically, the hidden layer activities are selectively reset to zero immediately before cue onset. The results demonstrate that despite the ablation, the model's

performance remained unaffected. This indicates that during the pre-cue delay period, information is stored within the Hebbian matrix, which aligns with the activity-silent strategy implemented during the training phase. Furthermore, these findings provide a foundation for addressing how item–rank conjunctive memories are organized within the fast Hebbian matrix and where they are stored.

B Three-Level Rank Recalling Tasks

B.1 Three-Level Rank Recalling Task Settings

To support deeper hierarchies, we introduce an additional phrase-chunking channel (Fig. S9) in Hebb-RNN and perform an additional study. When tested on a three-level task (e.g., syllables-words-phrases, each corresponding to Level 1,2 and 3). We find Hebb-RNN perform well (100.0% 0.0%, 4 seeds).

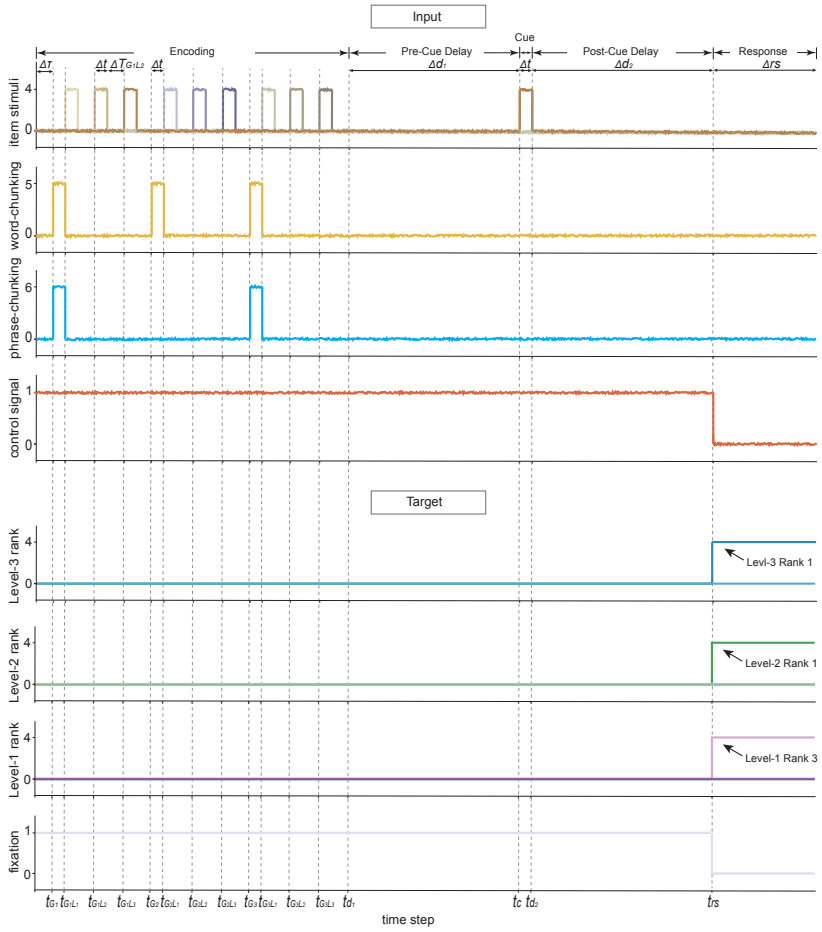

Figure S9: An Example of the Input and Target Sequences in the Three-Level Rank-Recall Working Memory Task. In this example, a [[3-3]-[3]] sequence is presented. The upper four panels depict the input channels, corresponding to the item stimuli, the word-chunking signal, the phrase-chunking signal, and the control signal, respectively. The lower panels show the output targets, including the Level-3 rank target, Level-2 rank target, Level-1 rank target, and fixation target. When the third cue is presented, the Level-1 rank is 3, Level-2 rank is 1, and Level-3 rank is 1, indicating that the cued item is on the third position in the first word of the first phrase.

B.2 Additional Results in Three-Level Rank Recalling Tasks

Initial visualizations reveal a regular geometry under these complex conditions. For example, in [[3-3]-[3]] sequences, each sequence comprises two phrases: the first phrase contains two tri-syllabic words, and the second phrase contains one tri-syllabic word. As shown in Fig. S10, the resulting geometry reflects the hierarchical organization of the sequence across three words: syllables belonging to the first word are represented by yellow points, those of the second word by blue points, and those of the third word by brown points. The first two words cluster together, forming the first phrase, while the third word occupies a distinct region corresponding to the second phrase.

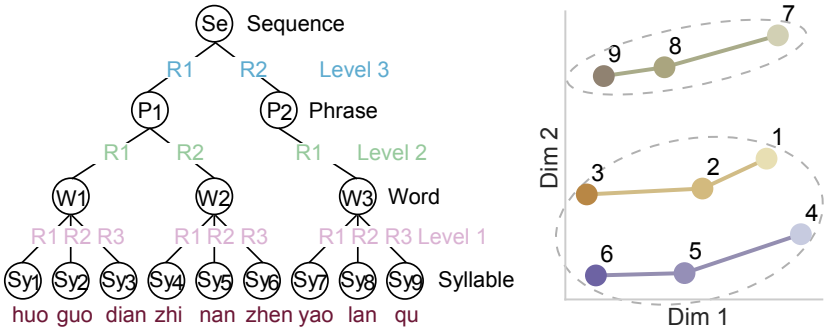

Figure S10: Three-Level Hierarchical Structure of [[3-3]-[3]] Sequences and the Underlying Geometry.

C Transfer Learning Task

C.1 Transfer Learning Task Settings

To investigate whether the learned internal representations generalize across stimulus formats, we perform a transfer learning procedure on networks pre-trained with one-hot encoded stimuli (Fig. 5E). A new set of 20 stimulus items is generated, each represented as a random vector drawn from a standard Gaussian distribution and ℓ_2-normalized, then scaled by the same stimulus intensity factor of 4, matching the number and the effective magnitude of the one-hot stimuli used during pre-training. Using this new stimulus set, we fine-tune only the input weight matrix $\boldsymbol{W}_{in}$ with learning rate of 0.001 while keeping all other learnable parameters fixed. The model architecture, configuration, training protocol, task paradigm, loss function, and evaluation protocol are identical to those described above.

To examine whether pre-training on one-hot stimuli accelerates subsequent learning on random stimuli, we train a separate set of networks from random initialization on the random stimulus set, with all learnable parameters free to update. Test accuracy is evaluated every 5 epochs for both the transfer learning networks and the from-scratch networks (Fig. 5F). For the transfer learning networks, epoch 0 corresponds to the pre-trained network before any fine-tuning on random stimuli; for the from-scratch networks, epoch 0 corresponds to randomly initialized parameters prior to training.

C.2 Additional Results in Transfer Learning Tasks

To investigate whether the network reuses the same representational template before and after transfer learning, we apply the low-dimensional projection method described in Appendix E.5 to compare representations across the two stimulus formats. Specifically, we first constructed the template of the pre-transfer network - trained on one-hot stimuli - for the 3-3-3 sequence-structure configuration, by averaging processed trajectories across all trials and projecting them onto the PCA subspace defined by the three principal dimensions (see Appendix E.5). We then project the 3-3-3 template of the post-transfer network - fine-tuned on random stimuli - onto the same PCA subspace derived from the pre-transfer network, enabling a direct geometric comparison of the two templates (Fig. S11).

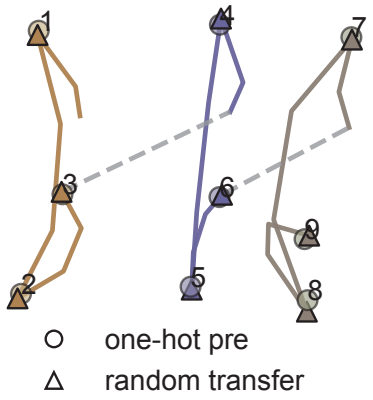

Figure S11: 2-D Geometry Before and After Transfer Learning.

D Naturalistic Memory Recall Task

D.1 Naturalistic Memory Recall Task Settings

To demonstrate that our task and stimulus design are non-trivial, we train networks using more naturalistic stimuli (Fig. 6A). Specifically, we select a set of real multisyllabic words used in Fan et al.'s paradigm [16], comprising 12 bi-syllabic, 12 tri-syllabic, and 12 quadri-syllabic words, and convert each syllable into a 768-dimensional embedding vector using the pretrained BERT-base-Chinese model (bert-base-chinese) available on Hugging Face, replacing the 20-dimensional one-hot stimuli used previously. New networks are trained from random initialization using these embedding vectors.

Dataset containing naturalistic stimuli differs from that containing the one-hot stimuli in several important respects. First, the embedding vectors carry semantic information, more closely reflecting real-world linguistic input. Second, the total number of distinct syllables — and thus embedding vectors — is 176, substantially larger than the 20 items used in the one-hot setting. Third, the ordering of syllables within each multisyllabic word is fixed, as dictated by lexical structure; since each sequence comprises three words spanning nine syllables in total, only the ordering of words within the sequence is randomized across trials. This stands in contrast to the one-hot stimulus dataset, in which all nine stimuli within a sequence are drawn and ordered entirely at random, with both chunk composition and within-chunk ordering varying independently across trials. Together, these properties make the naturalistic stimulus setting a more realistic and demanding test of the network's capacity for hierarchical sequence representation.

The training set is constructed such that every multisyllabic word appeared at least once. The test set is constructed following the same principle as described in Appendix A.3: training and test sets are non-overlapping in terms of the combination of words and their orderings within the sequence, such that word–ordering combinations in the test set are never seen during training, while all individual words have been encountered during training. All other aspects of the training and evaluation procedure are identical to those described previously.

We train Hebb-RNNs and GRUs on this dataset, with model configurations identical to those specified in Appendix A.2, and compare their test accuracy upon completion of training (Fig. 6E).

Moreover, as shown in Fig. 6F, hierarchical geometric representations consistently emerge in Hebb-RNNs trained on naturalistic stimuli (more details in Appendix E.1).

D.2 Additional Results in Naturalistic Memory Recall Tasks

We apply the same parametric RSA procedure to networks trained on the naturalistic memory recall tasks (see Appendix E.3). The best-fitting parameters similarly support a 2-D hierarchical geometry (Fig. S12). Bootstrap resampling ($N = 100$) with fixed LGS and fixed LGA respectively yielded

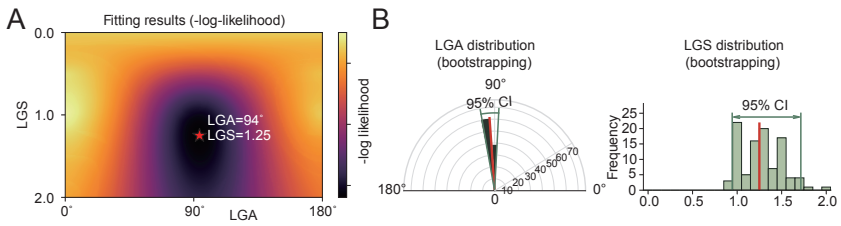

Figure S12: Model Fitting Performance as a Function of LGA and LGS Derived from a Parametric Representational Geometry Model, and Bootstrap Distributions of LGA (LGS Fixed at 1.25) and LGS (LGA Fixed at 94°), Each Based on 100 Bootstraps.

narrow 95% confidence intervals, further confirming the robustness of the estimated parameters and their consistency with the 2-D hierarchy hypothesis across stimulus formats.

D.3 Control Experiments Using Context-Independent Random Representations

To test whether prediction-based segmentation depends on semantic or contextual information in the BERT-derived representations, we constructed an additional control task using 50 items, each represented by a fixed random vector. Within each chunk, items followed a fixed order, whereas chunk order varied across sequences. To introduce overlap across chunks, each item had a probability of 0.2 of also appearing in another chunk.

We used the same hybrid architecture as in Fig. 6. The sensory-prediction RNN was trained to predict the next item, and a chunking signal was triggered when the prediction error exceeded 0.8. All other model and training settings followed the naturalistic memory recall task.

Under this context-independent setting, the Hebb-RNN achieved $100.0\% \pm 0.0\%$ test accuracy across four seeds, whereas the GRU baseline remained near chance at $10.0\% \pm 1.2\%$. These results show that prediction-based segmentation remains effective when item representations contain neither semantic nor contextual information.

E Analysis Pipeline and Visualization of the Hierarchical Geometry

We select the sequence-structure configuration of 3-3-3 for illustration, and categorize the trials into nine distinct groups with 10 trials sampled per group as mentioned before, and obtain the values of Hebbian matrix and hidden layer activities at each time step during each trial.

E.1 Visualization at Post-Cue Delay

We extract the hidden layer activities during the early post-cue delay period (specifically, a five-time-step window after cue withdrawal), and calculate each temporal rank representation by averaging activities over the 5 time steps and trials of the same group (Fig. 3). We then apply two dimensionality-reduction methods to obtain the hierarchical geometry. Firstly, following Fan et al., we compute the pairwise Euclidean distances between all points to obtain a 9×9 distance matrix (i.e., the empirical representational dissimilarity matrix, RDM), and apply classical multidimensional scaling (MDS) to project the empirical RDM onto a three-dimensional subspace defined by the first three MDS dimensions, in which points 1-9 represent nine ranks (Fig. 3C). We also apply principal component analysis (PCA) to the nine rank representation vectors and project them onto the subspace defined by the first three PC dimensions (Fig. 3A, D).

We further measure the principal angles between every two rank planes in Fig. 3E to quantify the geometric relationship. We first obtain an orthonormal basis for each plane using QR decomposition. We then compute the singular value decomposition (SVD) of the matrix formed by the inner products between the two bases. The plane angle is then defined as the smallest principal angle, obtained by taking the arccosine of the largest singular value.

E.2 Null Analysis of Low-Dimensional Variance Concentration

To assess whether the variance captured by the leading principal components exceeds that expected from high-dimensional random activity, we constructed two matched Gaussian null models: (1) a standard null that preserves the numbers of conditions and neural units, and (2) a trial-variance-matched null that additionally preserves heterogeneous per-unit trial-level variability.

For each model seed, we generated 10,000 independent null realizations and computed the cumulative variance explained by the first two principal components using the same PCA procedure as for the empirical representations. The observed rank representations showed approximately 80% variance explained by the first two PCs. In comparison, the standard Gaussian null yielded a mean of 30.7% (95% CI: [30.2%, 31.2%]), whereas the trial-variance-matched null yielded 38.8% (95% CI: [37.3%, 40.3%]). The observed variance concentration exceeded both null distributions (empirical $p < 10^{-4}$), confirming that the low-dimensional variance concentration is not an artifact of undersampling.

E.3 Parametric Representational Geometry Fitting

In Fig. 3F, Fig. S5 and Fig. S12, we fit a parametric representational geometry model with two free parameters, that is, the local–global angle (LGA) and the local–global scale (LGS), using the parametrical representational similarity analysis (RSA) approach as Fan et. al. The LGA denotes the angle between the local and global representational dimensions: a 0° LGA corresponds to the 1-D geometry (local and global ranks sharing a single dimension), whereas a non-zero LGA indicates a 2-D hierarchical geometry, with 90° denoting full orthogonality. The LGS denotes the ratio of representational distance between consecutive units along the local versus global dimensions.

Specifically, candidate RDMs are generated by sampling all combinations of LGA and LGS, and each is regressed against the empirical RDMs computed above (see Appendix E.1) via regression analysis (linear mixed model with fixed effects indicating across-seed effects and random effects indicating individual effects). The best-fitting parameters are identified as those minimizing the negative log-likelihood (Fig. 3F).

To assess the robustness of the best-fitting parameters, we perform bootstrap resampling ($N = 100$). Specifically, after identifying the best-fitting parameter pair (LGA, LGS), we fix LGS and resampled networks with replacement to re-estimate LGA across bootstrap iterations; the procedure is then repeated with LGA fixed to re-estimate LGS. The 95% confidence intervals are derived from the resulting bootstrap distributions (Fig. S5, Fig. S12).

E.4 Permutation Analysis of Global–Local Factorization

To assess whether the observed global–local plane organization exceeds that expected from the high-dimensional activity structure alone, we performed a trial-label permutation analysis across independently trained models. For each model, joint global–local rank labels were randomly permuted across trials while preserving the original hidden-state activity, trial variability, neural dimensionality, and covariance structure.

For each permutation, we recomputed the within-global, within-local, and between-global–local plane angles using the same procedure as in Fig. 3E. The observed angles were then compared with the permutation null distributions.

Table S5: Observed and permutation-null plane angles. Values are reported in degrees with 95% confidence intervals.

Plane-Angle Measure	**Observed**	**Permutation Null**	p **value**
Within global	11.68 [9.47, 14.80]	36.27 [31.39, 41.46]	0.0002
Within local	11.29 [10.22, 12.46]	36.28 [31.50, 41.42]	0.0002
Between global and local	73.93 [66.81, 80.09]	28.09 [24.64, 31.57]	0.0002

Compared with the permutation null, within-global and within-local planes were substantially more parallel, whereas global- and local-rank planes were more separated. All three effects were significant.

E.5 Visualization during Encoding

We extract the hidden layer activities during the whole encoding period and apply PCA on the neuronal dimension to obtain the first six PC dimensions (Fig. 4A). We specifically focus on the fourth, fifth, and sixth principal components (PCs), as these dimensions are more adept at capturing the shared abstract hierarchical structure underlying different trials and sequence-structure configurations.

To prepare the data for structural analysis, we first reduce the dimensionality of the hidden layer activities using these three PCs. We then segment the resulting trajectories into 13 distinct blocks, utilizing the boundaries demarcated by the dashed lines in the encoding phase of Fig. S1 (for 2-2-3 sequences, the total number of blocks is 13-2=11 and for 3-0-3 sequence, the total number of blocks is 11 as shown in Fig. S2). For example, blocks containing a chunking signal cover the full duration when a chunking signal is given, and blocks containing a stimulus input correspond to a short segment covering an item presentation and its subsequent brief interval before the next item. This segmentation allows for a precise temporal characterization of the neural dynamics as the encoding period progresses. Each trajectory block containing stimuli inputs is downsampled to 5 time steps and smoothed with a Gaussian kernel on time dimension for better visualization. Specifically, we apply a linear smoothing kernel with a window size of $w = 5$. To mitigate boundary effects associated with the convolution operation, each downsampled trajectory block is padded at both ends of the temporal axis by repeating the boundary values. The padding length at each end is set to $pad = w-1$, resulting in a padded sequence of total length $T' + 2w - 2$,

$$\mathbf{h}^k_{\text{padded}}(t) = \begin{cases} \mathbf{h}^k(1) & \text{if } t < 1 \\ \mathbf{h}^k(t) & \text{if } 1 \leq t \leq T' = 5 \\ \mathbf{h}^k(T') & \text{if } t > T' = 5 \end{cases} \tag{7}$$

where $t \in \{-pad + 1, -pad + 2, \cdots, 0, 1, \cdots, T', T' + 1, \cdots, T' + pad\}$. A 1D convolution is independently performed for each trial k and reduced hidden unit dimension i,

$$\hat{h}^{k,i}(t) = \sum_{p=0}^{w-1} \frac{1}{w} h^{k,i}_{\text{padded}}(t + p) \tag{8}$$

The resulting sequence after convolution has a length of $L = T' + w - 1$. To maintain consistency with the original trajectory duration, we perform a center crop to extract the sequence corresponding to the original length,

$$\{\mathbf{h}^k_{\text{smoothed}}(t)\}_{t=1}^{T'} = \{\hat{\mathbf{h}}^k(t)\}_{t=\lfloor \frac{L-T'}{2} \rfloor}^{\lfloor \frac{L+T'}{2} \rfloor} \tag{9}$$

To ensure temporal continuity between trajectories of different rank segments, we enforce the following boundary conditions,

$$\mathbf{h}^k_{\text{smoothed}}(1) = \mathbf{h}^k(1), \mathbf{h}^k_{\text{smoothed}}(T') = \mathbf{h}^k(T') \tag{10}$$

Finally, we plot the reduced and downsampled blocks with stimuli input for clarity. The dashed line connects the first time step of the next chunk and the last time step of the last chunk between two successive chunks.

We average these processed trajectories across all trials regardless of their group assignment, and then project them onto the subspace defined by these three dimensions, as our template. For each sequence-structure configuration of 3-3-3, 2-2-3 and 3-0-3, we project processed trajectory of one sample trial on the same subspace.

To compute the distance between template and trajectories of each sequence-structure configuration, we use the original trajectories without downsampling, smoothing or dimensional reduction (Fig. 4B). For example, when computing distance between template and 3-3-3 sequences, we first extract and match the blocks of template containing stimuli inputs with those of trajectory of each trial from 3-3-3 sequences. We then compute the distances of each matched block pairs and sum all the pairs

up. We finally normalize the overall distances by dividing them by the number of block pairs (for 3-3-3, the number of block pairs is 3+3+3=9; for 2-2-3, the number is 2+2+3=7 and for 3-0-3, the number is 3+0+3=6) and the number of time points of each block (that is, 12=$\Delta t + \Delta T$), thereby ensuring that distances between trajectories of varying lengths are commensurable. The distances are averaged across all trials of the same sequence-structure configuration. Furthermore, since trials can have different sequence lengths, aligning entire trajectories point by point is not feasible, and thus shuffled controls are computed by randomly permuting time points within each block of a trial trajectory.

E.6 Detailed Visualization of the Encoding Trajectory for a Sequence with a 3–0–3 Structure

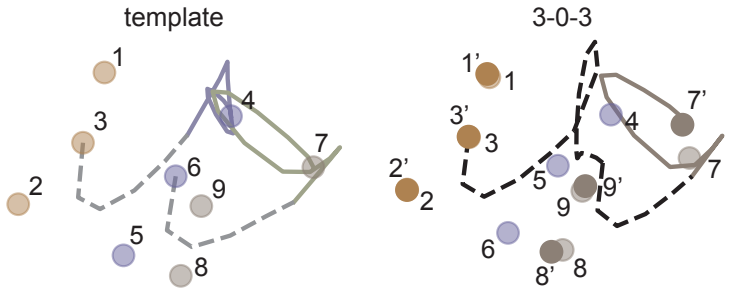

Figure S13: Detailed visualization of the encoding trajectory for a sequence with a 3–0–3 structure. Here, we explicitly illustrate how the representation evolves from the third item to the second chunk, and how it subsequently transitions from the second chunk to the seventh item in the third chunk. The left and right panels respectively display the template rank-geometry trajectory and the encoding trajectory segments for the 3–0–3 sequence, both corresponding to a specific portion of the encoding phase. Owing to the strong entanglement of the trajectory in the main visualization (that of 3-0-3 in Fig. 4A), the transition dynamics are not clearly resolved. Here, 0 denotes a condition in which only the chunking signal is presented, without any subsequent item inputs. Notably, even in the absence of following items, the chunking signal alone is sufficient to drive the rank representation in the rank geometry toward the position corresponding to the second chunk.

Fig. S13 provides a detailed and unprocessed visualization of how neural representations evolve from the third item of the first chunk to the second chunk, and how they subsequently transition from the second chunk to the seventh item in a sequence with a 3–0–3 structure. Specifically, for template rank-geometry trajectory, we select the blocks containing the second and third chunking signals, as well as the fourth and seventh item stimuli inputs, and project them onto the same subspace used in Fig. 4A without downsampling or smoothing. Likewise, for the encoding trajectory of 3-0-3 sequence, we select the blocks containing the second and third chunking signals, the seventh item stimuli inputs, as well as the empty chunks in Fig. S2 and project them onto the same subspace. The results show that, in such sequences, the chunking signal alone is sufficient to shift the network state along the abstract rank geometry from the first chunk to the second chunk, thereby enabling the subsequent computation of both global and local ranks. Together, these findings indicate that, within the hierarchical rank geometry, the chunking signal mainly supports global-rank computation.

E.7 Visualization for Hierarchical Geometry in Hebbian Matrix

We directly probe the fast Hebbian weights $A \odot P$ with diverse stimuli just before cue onset to reveal the hierarchical ordinal geometry in Fig. 5B. Specifically, we collect the Hebbian weights at the exact time step just before cue onset for each trial, as well as the input vectors at the exect cue onset (i.e., containing the cued item stimulus and control signal). Analogous to how a Hopfield network uses a probe vector to retrieve stored information from its synaptic weight matrix via iterative dynamics, we employ $(A \odot P) \cdot (W_{\text{in}} I)$ for each trial to obtain the retrieved vector through a single-step operation. We average the retrieved vectors across trials of the same group sharing the same sequential position of the cued item (as mentioned before) and obtain nine vectors representing the probed rank representation form Hebbian weights. We apply PCA to the nine vectors and project them onto the three-dimensional subspace defined by the first three PC dimensions.

F Ablation Study on Hebbian Weights

We do ablation studies on Hebbian weights as illustrated in Fig. 5C-D. Specifically, according to the updating rule in Eq. 2, the Hebbian matrix is updated additively within each trial. By defining the incremental update as $\Delta P(t) \coloneqq \eta \mathbf{h}(t)\mathbf{r}(t-1)^\top$, we have

$$P^k(t+1) = P^k(t) + \Delta P^k(t) = P^k(1) + \sum_{j=1}^{t} \Delta P^k(j) = \sum_{j=1}^{t} \Delta P^k(j) \tag{11}$$

Where k denotes each trial. At the onset of each trial, the Hebbian matrix is initialized to zero, i.e., $P^k(1) = \mathbf{0}$. By partitioning the trial into distinct phases (with notation corresponding to Fig. S1), the total Hebbian matrix at the final time step of the response phase, P^k_{total}, can be decomposed into a cumulative sum of component matrices,

$$\begin{aligned} P^k_{total} &= \sum_{j=1}^{T} \Delta P^k(j) \\ &= \sum_{j=1}^{t_{d_1}-1} \Delta P^k(j) + \sum_{j=t_{d_1}}^{t_c-1} \Delta P^k(j) + \sum_{j=t_c}^{t_{d_2}-1} \Delta P^k(j) + \sum_{j=t_{d_2}}^{t_{rs}-1} \Delta P^k(j) + \sum_{j=t_{rs}}^{T} \Delta P^k(j) \\ &\coloneqq P^k_{encoding} + P^k_{pre\text{-}cue} + P^k_{cue} + P^k_{post\text{-}cue} + P^k_{response} \end{aligned} \tag{12}$$

The term $P^k_{encoding}$ is further expanded to account for the hierarchical structure of the encoding phase,

$$P^k_{encoding} = P^k_{\Delta\tau} + P^k_{G_1} + P^k_{chunk_1} + P^k_{G_2} + P^k_{chunk_2} + P^k_{G_3} + P^k_{chunk_3} \tag{13}$$

Here, $P^k_{\Delta\tau}$ represents the cumulative update from the trial onset until the first chunking signal. $P^k_{G_i}$ covers the update during the full duration of the i-th chunking signal is given ($i = 1, 2, 3$) (i.e., blocks with chunking signal as mentioned in the former subsection). $P^k_{chunk_i}$ captures the update during the i-th chunk period, spanning from the stimulus onset of the first item to the end of the interval phase of the final item within that chunk.

Finally, $P^k_{chunk_i}$ can be further resolved into item-specific updates,

$$P^k_{chunk_i} = \sum_{j=1}^{V} P^k_{G_iL_j} \tag{14}$$

In this expression, $P^k_{G_iL_j}$ represents the cumulative update for the j-th item in the i-th chunk (i.e., the item with global rank i and local rank j), integrated from the stimulus onset to the end of the interval phase for that specific item (i.e., the blocks with stimuli inputs as mentioned in the former subsection). V denotes the total number of items contained within $chunk_i$.

To investigate how conjunctive memories are organized within the fast Hebbian matrix, we performed targeted ablations of individual item-rank components prior to cue onset in Fig. 5C. For each trial k, we selectively removed a given component G_iL_j by subtracting its specific plasticity contribution $P^k_{G_iL_j}$ from the Hebbian matrix at the precise time step immediately preceding cue onset, and utilize the result of subtraction, along with the corresponding hidden layer activities at the precise time step immediately preceding cue onset, as the new initialization state for the network. From this state onward, the network is driven by the remaining input sequence—comprising the cue, post-cue delay, and response phases (from t_c to the end of the trial).

Similarly, we performed targeted ablations of individual item-rank components during encoding period in Fig. 5D. For each trial k, we selectively removed a given component G_iL_j by subtracting its specific plasticity contribution $P^k_{G_iL_j}$ from the Hebbian matrix at the precise time step immediately

preceding the next item stimulus onset (or the next chunking signal onset, or the pre-cue delay onset), and utilize the result of subtraction, along with the corresponding hidden layer activities at the precise time step immediately preceding the next item stimulus onset (or the next chunking signal onset, or the pre-cue delay onset), as the new initialization state for the network. From this state onward, the network is driven by the remaining input sequence—comprising the remaining part of encoding, pre-cue delay, cue, post-cue delay, and response phases.

G Experimental Protocol and Model Configuration in Next-Stimulus Prediction Tasks

G.1 Next-Stimulus Prediction Task Settings

We present a scenario in which the chunking signal emerges naturally from prediction error. To this end, we designed a next-stimulus prediction task, in which the network is required to predict the upcoming stimulus during the inter-stimulus interval.

The task uses the multisyllabic words and corresponding BERT embedding vectors introduced in Appendix D.1. Unlike the Rank Recalling Task (Appendix A.1) and naturalistic memory recall task (Appendix D.1), the input to the network consists solely of the 768-dimensional embedding vectors, with no chunking signal or control signal channels. The output layer is 176-dimensional, corresponding to the total number of distinct syllables, and the network is trained to predict the identity of the next syllable.

The stimulus timing structure is similar to the encoding period of the Rank Recalling Task. Each syllable is presented for a fixed duration of $\Delta t = 5$, with each embedding vector scaled by a stimulus intensity factor of 4 upon presentation. The inter-stimulus intervals — including the interval preceding the first syllable — are randomly sampled from a uniform distribution $\Delta T \sim \mathcal{U}[5, 12]$. Unlike the Rank Recalling Task, the number of syllables and words per trial varies across trials, ranging from two bi-syllabic words (4 syllables in total) to four quadri-syllabic words (16 syllables in total). The network is trained to generate predictions throughout all inter-stimulus intervals, from the interval preceding the first syllable (predicting the first syllable of the first word) up to and including the interval preceding the final syllable.

Without access to any explicit chunking signal, the network must learn the transitional probabilities between syllables and discover the underlying lexical structure - that is, the identities of the multisyllabic words - from the continuous speech stream. Word boundaries are expected to manifest as elevated prediction error at word-initial positions relative to word-internal transitions, and this prediction error signal serves as the basis for the emergent chunking signal in our framework.

G.2 Model and Training Details for the Next-Stimulus Prediction Task

We train a vanilla RNN with 300 hidden units on this task. The loss function is the mean squared error (MSE) computed exclusively during inter-stimulus intervals, with no loss applied during stimulus presentation periods (Fig. 6C).

G.3 Prediction Error Computing

After training, the network's output is first passed through a softmax function to obtain a probability distribution over all possible next syllables. The prediction error at each time step is then computed as $1 - p(\text{correct next syllable})$, reflecting the network's uncertainty about the upcoming item (Fig. 6D).

H Perturbation Analysis of Global-Local Representational Balance

To further characterize the global–local representational balance in the Hebb-RNN, we performed two perturbation analyses by independently varying (i) the chunk-boundary input amplitude and (ii) the local-rank target amplitude. All other model and training parameters were kept identical to those used in Fig. 3. The default setting used a chunk-input amplitude of 5 and a local-rank target amplitude of 4.

Table S6: LGS across different chunk-boundary input amplitudes. All other model and training parameters were kept identical to those used in Fig. 3.

Chunk-Input Amplitude	LGS
4	1.74
5	1.39
6	1.07
7	1.09
8	0.81

Table S7: LGS across different local-rank target amplitudes. All other model and training parameters were kept identical to those used in Fig. 3.

Local-Rank Target Amplitude	LGS
2	0.65
4	1.39
6	2.00

We first varied the chunk-boundary input amplitude from 4 to 8 and computed the corresponding LGS values.

We next varied the local-rank target amplitude from 2 to 6 and computed the corresponding LGS values.

Across these perturbations, LGS was bidirectionally modulated: stronger chunk-boundary inputs were accompanied by lower LGS, whereas stronger local-rank targets were accompanied by higher LGS.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the contributions, which align with the experimental content in Sections 2–3.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: This work is empirical and does not contain formal theorems or proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Sections 2-3 and Appendix provide detailed model, hyperparameter settings, training and testing settings to support reproducibility.

   Guidelines:

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

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

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

Answer: [No]

Justification: Code is not released due to ongoing internal review, but all technical details are described in the paper and appendix to facilitate reproducibility.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We report all the training and test details as well as other settings in Sections 2-3 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report mean ± standard deviation across random seeds for performance comparisons (e.g., Fig. 2F), and assess the statistical robustness of the main representational-geometry results using matched-null analyses, permutation tests, and bootstrap confidence intervals (Sec. 3.2 and Appendix E).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: All experiments are conducted on a personal desktop equipped with 32 GB RAM and a single NVIDIA RTX 4080 GPU; runtimes are several hours per run.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our research fully adheres to the NeurIPS Code of Ethics and involves no sensitive or identifiable data.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This is a foundational study without direct societal deployment, so broader impacts are not applicable.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We do not release new models or data requiring safeguards.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We use data from Fan et al., which is cited where used in the paper, and the pretrained BERT-base-Chinese model (bert-base-chinese) from Hugging Face, licensed under Apache 2.0. The dataset from Fan et al. is publicly available but has no explicit license; we have obtained permission from the original authors to use it for academic research purposes.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

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

13. **New assets**

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

    Answer: [N/A]

    Justification: No new assets are introduced; only standard models are used.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: No human subjects or crowdsourcing were involved in this study.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: As no human subjects were involved, IRB approval was not needed.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were only used for writing polish; core methods and experiments were developed independently.

    Guidelines:

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