
Shaping Sequence Attractor Schema in Recurrent Neural Networks - Technical Appendices

A Training and Model Specifications

A.1 Odor-Sequence Task: Input and Target Functions

Fig. S1 illustrates examples of input and target sequences in task primitive learning (a), task sequence learning (b), and task schema learning (c).

A.2 Model Parameters

For the RNNs shown in Fig.2–4, the RNNs consist of $N = 200$ units, with a time constant $\tau = 1$, and all initial states are set to zero. The input-to-recurrent weights are independently initialized using a Gaussian distribution with mean zero and variance g^2/\sqrt{N} , where the scaling factor is $g = 0.1$. The networks are trained using the ADAM optimizer, with a learning rate of 0.001 for task primitive learning and task sequence learning, and 0.0001 for task schema learning. The training batch size is fixed at 8 for all experiments. In the experimental setup, it did not explicitly require rats to report odor identity; their task was confined to making binary choices. However, we incorporate identity prediction into our Task Primitive Learning based on compelling neurophysiological evidence and model rationale: (1) Our RNNs exclusively models the OFC, which likely receives necessary supervised signals from other interconnected cortical regions; (2) Experimental data show that OFC neurons exhibit odor selectivity when odor identity is behaviorally relevant, and this selectivity diminishes when it is not (e.g., when identity is irrelevant to the reward outcome) [2]. In task primitive learning, the odor identity is essential as it directly aids reward prediction. Conversely, in the subsequent stages (task sequence and schema learning), the reward outcome becomes dependent primarily on the sequence context rather than the odor identity itself; consequently, the model’s need to predict identity is removed. All experiments were run on an Intel i9-14900K CPU with 32GB RAM, with each training session lasting around 30 minutes.

In Fig.2d–e present RDMs of the RNNs after task schema learning. During this phase, five tasks are used, consistent with the experimental settings described in the main text. For RDM computation, noise with a mean of zero and a variance of 1.5. In Fig.3d and Fig. S3, for better visualizing neural trajectories, no background noise is added, and the output weights are frozen and non-trainable.

In Fig. 4C and Fig. S4, to ensure a fair comparison of learning efficiency, we include the number of training batches (also referred to as trials) used during task primitive and sequence learning when calculating the total trials to criterion for task schema learning.

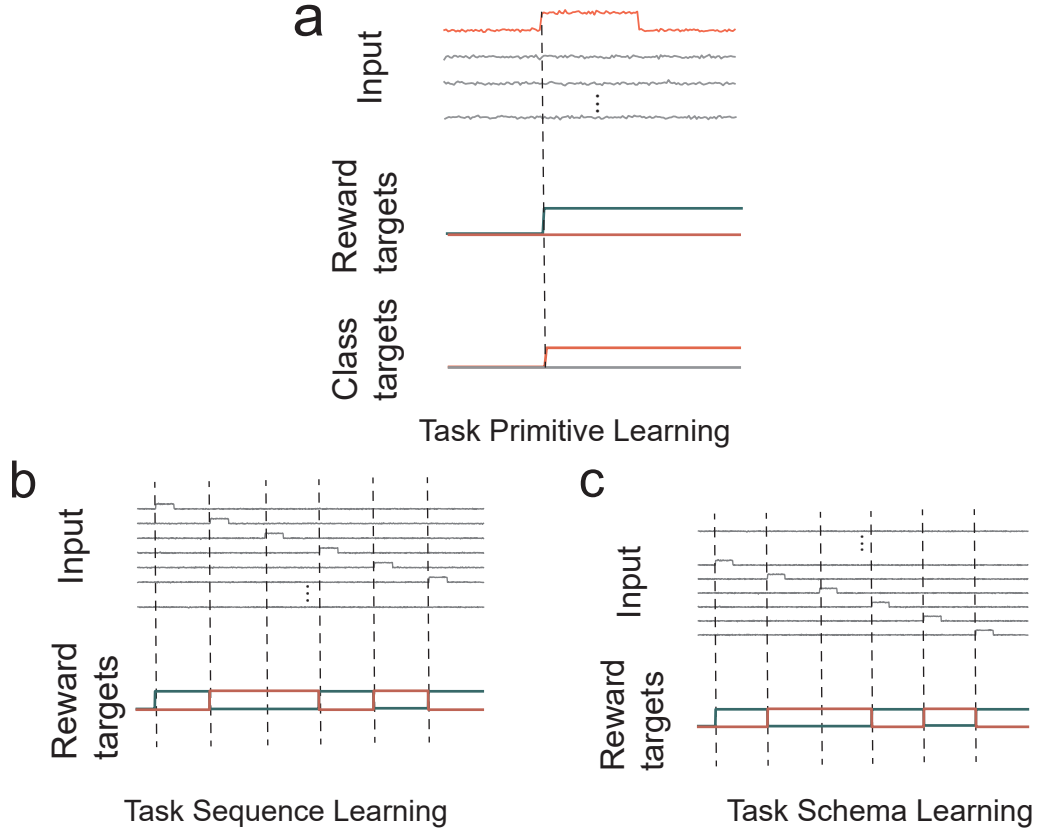


Figure S1: Examples of input and target sequences during task primitive learning (a), task sequence learning (b), and task schema learning (c). In reward targets, blue represents non-reward, and red denotes reward. All sequences are corrupted by Gaussian white noise with a mean of zero and a variance of one. (a) In task primitive learning, inputs are 96-dimensional sequences, structured as 5 steps of delay, 5 steps of stimuli, and 5 steps of delay. The stimulus is a one-hot vector, where the active element has a value of 20. Targets are divided into two components: a 2-dimensional reward target and a 16-dimensional class target. Both targets utilize ramping functions, meaning that upon stimulus onset, the corresponding decision unit in both the reward and class targets transitions to one, while others remain zero. (b) In task sequence learning, each input sequence consists of six equally spaced odor stimuli, each presented for 5 steps, totaling a sequence length of 90. RNNs are trained to predict only rewards, resulting in 2-dimensional outputs. The reward targets switch either 0 or 2 upon the onset of each odor stimulus. (c) In task schema learning, the input and target sequences are similar to those in task sequence learning (b), with the exception that odor stimuli consist of a new set of odors for each problem.

B Additional Results

B.1 Population Neural Activity Clustering Analysis

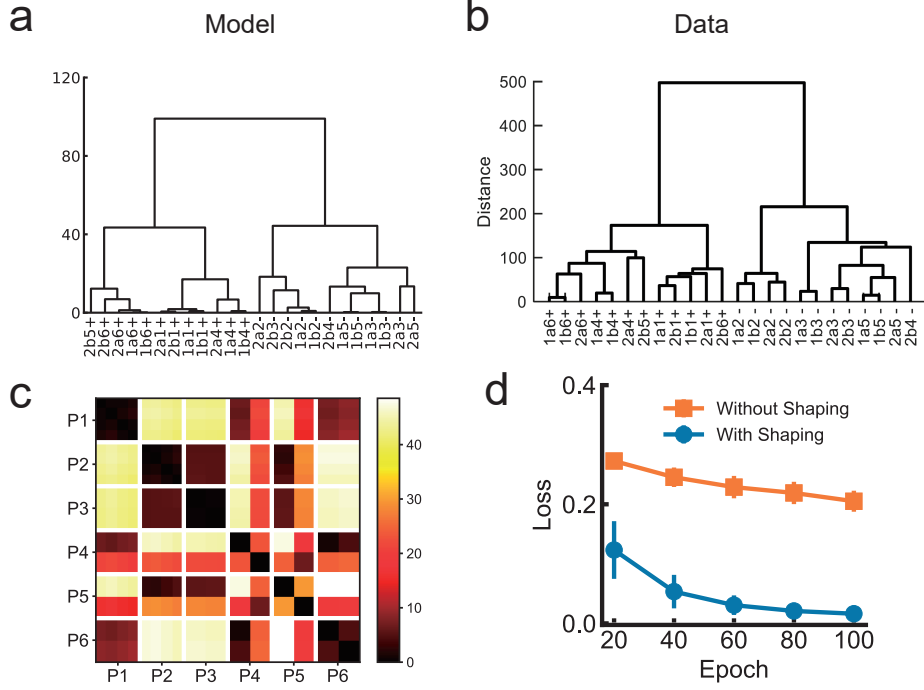


Figure S2: Hierarchical clustering is performed on 24 trial types based on population neural activity. In (a), data from RNNs are clustered. The dissimilarity matrix shown in Figure 2d of the main text is used to generate the clustering tree via the unweighted average linkage method, with the resulting structure illustrated as a dendrogram. In (b), the hierarchical clustering results for rat OFC ensembles are presented, adapted from [1]. (c) RDM of unshaped RNN hidden activities. (d) Training losses of shaped and unshaped RNNs in problem 1.

Based on the representational dissimilarity matrices D , as shown in Fig.2d of the main text, we construct a hierarchical clustering tree using the average linkage method. This method quantifies the dissimilarity between two clusters by computing the average distance across all pairs of samples from the two clusters. The distance between clusters C_i and C_j is defined as:

$$d(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y), \quad (1)$$

where C_i and C_j denote the two clusters, $|C_i|$ and $|C_j|$ are the number of samples in each cluster, and $d(x, y)$ represents the distance between samples x and y . Based on this distance metric, the algorithm iteratively merges the pair of clusters with the smallest inter-cluster distance, ultimately constructing a complete hierarchical clustering tree (dendrogram), as shown in Fig. S2a.

As illustrated in Fig. S2, dendrograms generated under 24 position-specific conditions (4 trial types \times 6 spatial positions) for the RNNs reveal a hierarchical representation structure that closely mirrors that observed in the experimental data. First, trial types are grouped into two major branches based on the presence or absence of reward. Each major branch is then further divided into two sub-branches according to spatial position. Within each position, trials with the same stimulus type form tightly clustered subgroups, reflecting minimal intra-group distance.

The close correspondence between the dendrogram structures derived from the model and the experimental data suggests that the RNN successfully captures the hierarchical task representation structure and exhibits a representational geometry similar to OFC.

B.2 Dimensionality Reduction and Visualization of Neural Dynamics

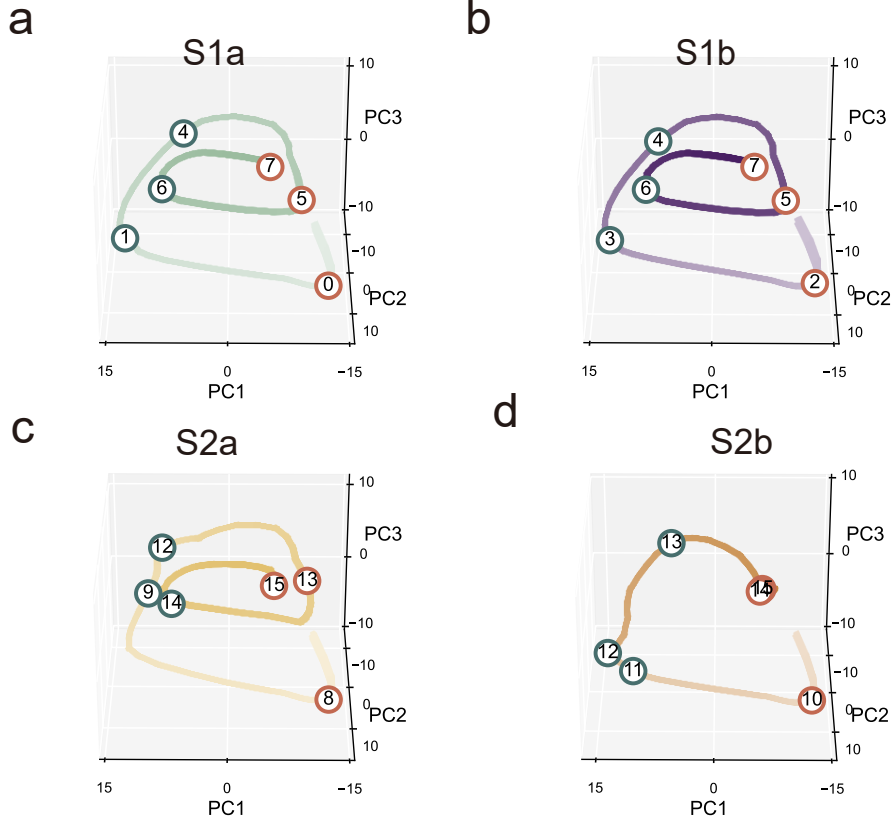


Figure S3: PCA visualization of neural trajectories for four sequences within the shaped RNN. The trajectories for sequences (a) S1a, (b) S1b, (c) S2a, and (d) S2b are projected onto their top three principal components. Additionally, six fixed points per sequence, identified via an optimization-based analysis, are explicitly marked.

We use PCA to reduce the dimensionality of the population neural activities of RNNs after shaping. Specifically, we generate $N_s = 80$ population activity trajectories, with 20 samples for each of the four sequence types: S1a, S1b, S2a, and S2b. Each trajectory spans $T = 95$ time steps. We record the hidden unit activities and construct a neural activity tensor:

$$R \in \mathbb{R}^{N \times N_s \times T}, \quad (2)$$

where $N = 200$ is the number of neurons in the RNN.

To perform dimensionality reduction, we first flatten the tensor R into a matrix:

$$R' \in \mathbb{R}^{N \times (N_s \times T)}, \quad (3)$$

by concatenating all trajectories along the time dimension, yielding an $N \times N_s T$ matrix. We then standardize the data by subtracting the mean across all samples and compute the covariance matrix:

$$C = \frac{1}{N_s T - 1} (R' - \langle R' \rangle) (R' - \langle R' \rangle)^\top, \quad (4)$$

where $\langle R' \rangle$ denotes the mean activity vector across all samples. Next, we perform eigendecomposition of the covariance matrix C and extract the top three principal components. The original neural activity is projected into this reduced-dimensional space to obtain:

$$R^{\text{PCA}} \in \mathbb{R}^{k \times N_s \times T}, \quad (5)$$

where $k = 3$ is the number of principal components retained. To visualize neural dynamics, we compute the average trajectory for each of the four sequence types by averaging over the 20 corresponding samples at each time step:

$$\bar{R}_s^{\text{PCA}}(t) = \frac{1}{20} \sum_{i \in \mathcal{S}_s} R_{:,i,t}^{\text{PCA}}, \quad s = 1, \dots, 4, \quad (6)$$

where \mathcal{S}_s denotes the sequence set corresponding to the s -th sequence type. The resulting averaged trajectories $\bar{R}_s^{\text{PCA}}(t)$ are visualized using the top three principal components to illustrate the temporal evolution of neural representations for better visualization (see Fig. S3).

B.3 Perturbation Analysis of Model Dynamics

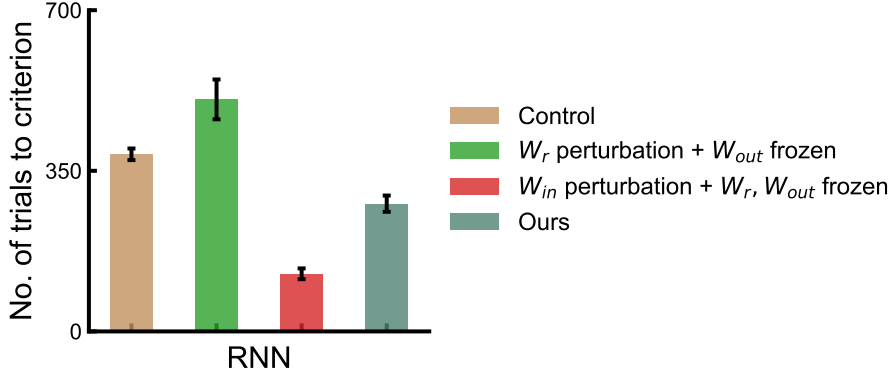


Figure S4: Perturbation analysis of RNNs with shaping to assess the role of sequence attractor reuse during task schema learning. An RNN is first trained through shaping and completes four problems during task schema learning. It is then trained on a fifth problem under different perturbation conditions to evaluate how the reuse of internal dynamics (sequence attractors) contributes to learning efficiency. The experimental conditions are as follows: Control: All synaptic weight matrices in the shaped RNN are perturbed by randomly shuffling their values. W_r perturbation + W_{out} frozen: The recurrent weights (W_r) are randomly shuffled and learnable, while W_{out} is frozen. W_{in} remains learnable. W_{in} perturbation + W_r, W_{out} frozen: Only the input-to-recurrent connections (W_{in}) are randomly shuffled and learnable, while the recurrent (W_r) and output (W_{out}) connections are frozen. Ours: No weight perturbation is applied. The evaluation criterion is the number of training trials (or batches) required for the training loss on the fifth task to reach a threshold of 0.05. To ensure a fair comparison, the learning rate is set to 0.001 for the Control condition, W_{in} perturbation + W_r, W_{out} frozen and W_r perturbation + W_{out} frozen, while a smaller learning rate of 0.0001 was used in our approach. Results are averaged over five random seeds.

As shown in Fig. 3 and Sec. 4.2 of the main text, four distinct sequence attractors emerge in the RNN's hidden state space after shaping. During task schema learning, the RNN also exhibits a learning-to-learn effect. This raises the question: do these sequence attractors contribute to faster learning of new tasks by serving as internal schemas? To investigate this, we first train an RNN via shaping. And after completing four problems in task schema learning, we then assess its learning efficiency on a fifth, novel odor-sequence task under different perturbation conditions. As shown in Fig. S4, our model learns the new task significantly faster than the control model, where all connection weights are randomly shuffled, eliminating prior knowledge. We further test a condition where only the input weights (W_{in}) are learnable, while the recurrent (W_r) and output (W_{out}) weights are frozen. If the sequence attractors function as reusable schemas, the network only needs to simply bind new input stimuli to the existing attractors, thereby enabling rapid learning. Indeed, the " W_{in} perturbation" model requires far fewer training trials to reach the loss threshold, supporting this hypothesis. Finally, we perturb W_r - thereby disrupting the internal attractor dynamics - while keeping W_{out} frozen. In this condition, the model requires substantially more training trials, further confirming that intact sequence attractors play a key role in accelerating learning of new tasks.

B.4 Shaping Enhances RDM Correlation in LSTM and GRU Architectures

Network	RDMs Correlation
LSTM with shaping	0.89 ± 0.01
LSTM without shaping	0.87 ± 0.03
GRU with shaping	0.906 ± 0.01
GRU without shaping	0.87 ± 0.007
RNN with shaping	0.902 ± 0.02

Table S1: RDM comparison of LSTM and GRU, averaged over six seeds.

To assess the architectural dependence of attractor formation, we conducted additional experiments using LSTM and GRU networks with 200 recurrent units, trained under the same shaping condition as the vanilla RNN. As shown in Tab. S1, both gated architectures achieve comparable representational similarity to neural data.

B.5 Training Paradigm Comparison

We conduct additional control experiments, summarized in Tab. S2. Both the alternating and reverse protocols significantly improve learning compared to no shaping. Notably, the alternating protocol outperform standard shaping, likely because it facilitates the acquisition of both primitive- and sequence-level attractors. These results can potentially lead to improvements in the efficiency of animal training and experimental shaping. Additionally, we also conduct a control experiment where the model in Stage 1 was trained only to predict reward, without odor identity (Control model), shown in Tab. S3. While the control model improves learning efficiency over the no-shaping baseline, it is still less effective than our full model that includes odor identity prediction. This suggests that explicitly training odor identity helps the network develop richer primitive attractors, which in turn facilitate sequence learning in later stages.

Table S2: Training Paradigm Comparison. L1: Task Primitive Learning; L2: Task Sequence Learning; L3: Task Schema Learning. All results represent the mean performance over six random seeds using the same training epochs.

Training Paradigm	Trials to criterion (Problem 1)
Our shaping (L1-L2-L3)	964.6 ± 110
Alternating (L1-L2-L1-L2-L3)	857.6 ± 21
Reverse (L2-L1-L3)	1463 ± 54
No shaping	1725.5 ± 75

Table S3: Odor Identity Prediction in Task Primitive Phase Accelerates Schema Learning. Efficiency is measured by the number of training trials to criterion (Problem 1). The "Control" model predicts only rewards, and the "Ours" model predicts both rewards and odor identity. Results are averaged over four seeds.

Model	Trials to criterion (P1)
RNN without shaping	1725.5 ± 75
Control	1259.33 ± 130.3
RNN with shaping	964.6 ± 110

B.6 Parameter Robustness of RNNs with Shaping

We vary the input and output magnitudes and find that the network performed robustly across a wide range of parameters (as shown in Tab. S4 and Tab. S5). We also observe that the shaping protocol maintains its effectiveness even with higher learning rates (e.g., $99.01 \pm 2.7\%$ accuracy, $\text{RDM}=0.90$ at 5×10^{-4}), although excessively large rates could disrupt existing attractors.

Table S4: Performance with Fixed Desired Output (Value = 2). All results are averaged to 4 seeds.

Input	Accuracy (%)	RDMs Correlation
15	99.39 ± 0.037	0.894 ± 0.018
20	99.89 ± 0.034	0.902 ± 0.021
25	99.97 ± 0.043	0.897 ± 0.004

Table S5: Performance with Fixed Non-Zero Input (Value = 20). All results are averaged to 4 seeds.

Target	Accuracy (%)	RDMs Correlation
1	99.23 ± 0.098	0.874 ± 0.013
2	99.89 ± 0.034	0.902 ± 0.021
3	99.08 ± 2.366	0.896 ± 0.007

C Analysis and Visualization of RNN Dynamics

C.1 Computation of the Representational Dissimilarity Matrix (RDM)

To evaluate the representational structure of the network, we follow the method in [1] and construct a data tensor $R \in \mathbb{R}^{N \times N_{\text{trial}} \times T_r}$ to represent trial types across different sequences. As described in Sec. B.2 of the supplementary text, $N = 200$ is the number of units in the RNN. Each of the four sequences (S1a, S1b, S2a, and S2b) contains 6 positions, with 20 sample trials per position, resulting in a total of $N_{\text{trial}} = 20 \times 24 = 480$ trials. The neural activity for each trial is defined as the response from the onset of the odor (lasting 5 time steps) followed by a delay period of 10 time steps, resulting in a total duration of $T = 5 + 10 = 15$ time steps. This yields the final activity tensor:

$$R \in \mathbb{R}^{N \times N_{\text{trial}} \times T}. \quad (7)$$

To compute the RDM, we first average the neural activity over time for each trial:

$$\bar{R}_{n,m} = \frac{1}{T} \sum_{t=1}^T R_{n,m,t}, \quad \bar{R} \in \mathbb{R}^{N \times N_{\text{trial}}}. \quad (8)$$

Next, we perform PCA on the time-averaged data. The covariance matrix is computed as:

$$C = \frac{1}{N_{\text{trial}}} \bar{R} \bar{R}^\top \in \mathbb{R}^{N \times N}. \quad (9)$$

We apply eigendecomposition to C and project the data onto the top $k = 5$ principal components:

$$R^{\text{PCA}} \in \mathbb{R}^{k \times N_{\text{trial}}}. \quad (10)$$

We then average the PCA-transformed representations across the 20 samples of each of the 24 trial types to obtain:

$$R^{\text{mean}} \in \mathbb{R}^{k \times 24}. \quad (11)$$

Finally, we compute the pairwise Euclidean distances between the mean representations of each trial type using the top 5 principal components:

$$D_{i,j} = \|R_{:,i}^{\text{mean}} - R_{:,j}^{\text{mean}}\|_2, \quad i, j \in \{1, \dots, 24\}. \quad (12)$$

The resulting distance matrix $D \in \mathbb{R}^{24 \times 24}$ serves as the RDM, characterizing the geometric structure of neural representations across trial types.

C.2 Fixed Points Analysis

To reveal the underlying structure of the RNN’s state space after shaping, we investigate the fixed points of the network dynamics and assess their stability. This analysis sheds light on how the network organizes computation through its dynamic landscape.

Identifying Fixed and Slow Points. Fixed points - and more generally, slow points - provide key insights into the computational roles of specific states in an RNN. A fixed point is a stable configuration of neural activity that remains unchanged under constant input, i.e., $\mathbf{r}^* = F(\mathbf{r}^*, \mathbf{I})$, where $F(\cdot)$ is the RNN’s update function. A slow point, while not strictly stationary, changes only minimally under the same conditions, i.e., $\mathbf{r}^* \approx F(\mathbf{r}^*, \mathbf{I})$. The dynamics of the RNN are governed by the continuous-time equation:

$$\tau \frac{d\mathbf{x}(t)}{dt} = -\mathbf{x}(t) + \mathbf{W}^{\text{rec}}\mathbf{r}(t) + \mathbf{W}^{\text{inp}}\mathbf{I}(t) + \mathbf{I}, \quad (13)$$

which we approximate using first-order Euler integration with a step size $\Delta t = 1$. The corresponding discrete-time update rule becomes:

$$\begin{aligned} \mathbf{r}(t+1) &= F(\mathbf{r}(t), \mathbf{I}(t)) \\ &= \tanh \left(\left(1 - \frac{\Delta t}{\tau} \right) \arctan(\mathbf{r}(t)) + \frac{\Delta t}{\tau} (\mathbf{W}^{\text{rec}}\mathbf{r}(t) + \mathbf{W}^{\text{inp}}\mathbf{I}(t) + \mathbf{I}) \right) \end{aligned} \quad (14)$$

To locate fixed or slow points, we minimize the quantity:

$$q = \frac{1}{2} \|F(\mathbf{r}, \mathbf{I} = \mathbf{0}) - \mathbf{r}\|^2, \quad (15)$$

which measures how much the state \mathbf{r} changes in the absence of external input. We follow the method introduced by Sussillo and Barak [3], initializing from a variety of network states observed during training and using backpropagation through time (BPTT) to minimize q . States for which $q < 10^{-7}$ are considered fixed points. In the odor-sequence task (see Fig.3 in the main text), this procedure identifies 6 fixed points in each sequence of the learned RNN dynamics.

Stability Analysis of Fixed Points. To further understand the role of each fixed point, we assess their stability by analyzing how the network responds to small perturbations. We apply small deviations $\Delta\mathbf{r}(t)$ and $\Delta\mathbf{I}(t)$ to the fixed point \mathbf{r}^* and examine how these perturbations evolve:

$$\mathbf{r}(t+1) = F(\mathbf{r}^* + \Delta\mathbf{r}(t), \mathbf{I}^* + \Delta\mathbf{I}(t)) \approx F(\mathbf{r}^*, \mathbf{I}^*) + \mathbf{J}^{\text{rec}}\Delta\mathbf{r}(t) + \mathbf{J}^{\text{inp}}\Delta\mathbf{I}(t), \quad (16)$$

where \mathbf{J}^{rec} and \mathbf{J}^{inp} are the Jacobians of the update function with respect to the recurrent state and the input:

$$J_{ij}^{\text{rec}}(\mathbf{r}^*, \mathbf{I}^*) = \left. \frac{\partial F_i(\mathbf{r}, \mathbf{I})}{\partial r_j} \right|_{(\mathbf{r}^*, \mathbf{I}^*)}, \quad J_{ij}^{\text{inp}}(\mathbf{r}^*, \mathbf{I}^*) = \left. \frac{\partial F_i(\mathbf{r}, \mathbf{I})}{\partial I_j} \right|_{(\mathbf{r}^*, \mathbf{I}^*)}. \quad (17)$$

This linearized system enables us to characterize the local stability of each fixed point. Eigenvalues of $\mathbf{J}^{\text{rec}}(\mathbf{r}^*, \mathbf{I}^*)$ indicate whether perturbations grow or decay over time - eigenvalues within the unit circle correspond to stable directions, while those outside indicate instability. This analysis provides mechanistic insights into how fixed points support memory, decision boundaries, or transitions between task states in the network.

C.3 Tracking the Evolution of Prior Attractors

To examine how the stable states (i.e., attractors) of the RNN evolve during task sequence learning, we adopt an attractor tracking procedure, illustrated here using the transition of attractors associated with odor 1 as an example.

First, during the *task primitive learning* phase, we extract stable attractor states from the trained RNN, such as \mathbf{r}_0 , which corresponds to the odor 1 input condition. These states are used as initial points for tracking in the subsequent *task sequence learning* phase.

Second, during task sequence learning, we probe the evolution of these attractors at regular intervals. Specifically, after every $n = 4$ training epochs, we initialize the RNN’s hidden state with the attractor state obtained from the previous interval (e.g., \mathbf{r}_{i-1}), then allow the network to evolve without external input until it converges to a new stable state, denoted as \mathbf{r}_i . This process is repeated over the course of training, producing a sequence of evolving attractor states:

$$\{\mathbf{r}_0, \mathbf{r}_1, \dots, \mathbf{r}_{T_n}\},$$

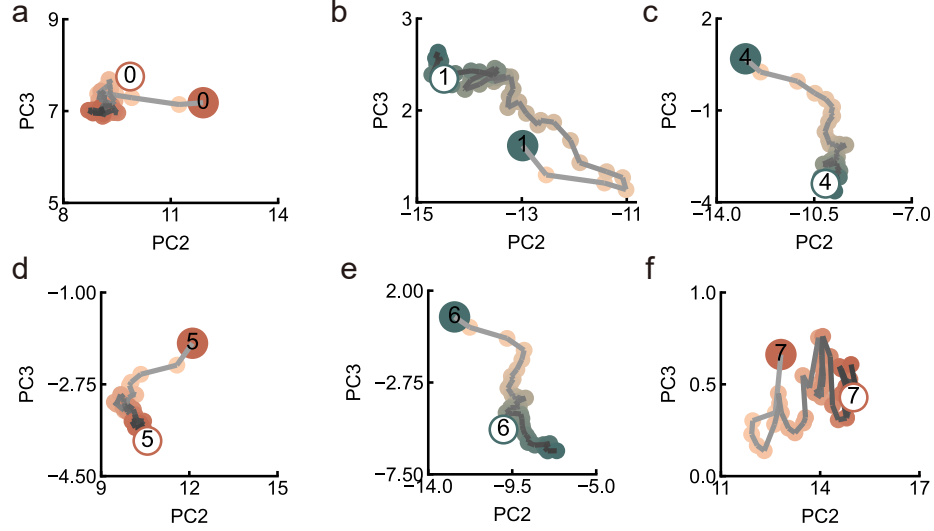


Figure S5: PCA visualization of attractor evolution dynamics during task sequence learning. The trajectories of attractor states are visualized in the PCA space to illustrate their evolution. Solid circles represent the attractor states identified after the *task primitive learning* phase, while hollow circles indicate the corresponding attractors for the S1a sequence after *task sequence learning*. Blue markers denote non-rewarded odors; red markers indicate rewarded odors. Panels (a)–(f) show the attractor evolution corresponding to different odors. In most cases, the prior attractors successfully evolve into their target states within the S1a sequence (odor 0(a), 1(b), 4(c), 5(d), and 7(f)). However, the attractor associated with odor 6 (e) fails to reach the correct state.

which we refer to as the attractor evolution trajectory.

Third, upon completion of training, we independently identify the attractor $\hat{\mathbf{r}}_{T_n}$ associated with the same odor (e.g., odor 1 in the S1a sequence) using an optimization-based fixed-point solver. By comparing the final evolved state \mathbf{r}_{T_n} with the converged attractor $\hat{\mathbf{r}}_{T_n}$, we assess the success of attractor reuse via their Euclidean distance.

This approach relies on the observation that RNN weights change incrementally with each training update. If the interval between tracking steps is sufficiently small, the dynamics of attractor evolution can be approximated by a continuous trajectory in the hidden state space, allowing us to capture fine-grained transitions and assess whether prior attractors are preserved or repurposed.

As shown in Fig. S5, multiple examples of evolving attractors from the S1a sequence are visualized. All prior attractors, except the one corresponding to odor index 6, successfully evolve into their corresponding task-specific attractors after training. The single failure case may result from stochastic noise during training, such as variability introduced by stochastic gradient descent.

C.4 Comparison with Experimental Data Using CCA

If the RNN forms a schema in a low-dimensional subspace, the true neural representation should become increasingly similar across problems and models as learning progresses. To investigate this in our model, we perform Canonical Correlation Analysis (CCA) on neural activities in RNNs trained via shaping.

In the context of task schema learning, we first apply PCA, as detailed in Sec. C.1, to obtain low-dimensional neural representations, denoted as R^{PCA} . Subsequently, we use CCA to align the neural manifold of activities across different models (cross-subject) or different problems (cross-problem). CCA identifies linear projections of paired datasets that maximize their correlation. The optimization

problem is defined as:

$$\max_{\mathbf{w}_X, \mathbf{w}_Y} \text{corr}(\mathbf{X}\mathbf{w}_X, \mathbf{Y}\mathbf{w}_Y) = \frac{\mathbf{w}_X^\top \Sigma_{XY} \mathbf{w}_Y}{\sqrt{\mathbf{w}_X^\top \Sigma_{XX} \mathbf{w}_X} \sqrt{\mathbf{w}_Y^\top \Sigma_{YY} \mathbf{w}_Y}}, \quad (18)$$

where $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{k \times M}$ represent the PCA-reduced neural activity data under different conditions. The resulting projections $U = \mathbf{X}\mathbf{W}_X$ and $V = \mathbf{Y}\mathbf{W}_Y$ reflect the shared representational structure between paired data.

The CCA projections $U \in \mathbb{R}^{T \times k}$ are then reshaped into (T_c, T_t, k) , where $T_c = 24$ corresponds to unique position-trial type combinations (4 trial types \times 6 positions), $T_t = 20$ is the number of trials per condition, and k is the number of canonical components. We compute the mean and standard deviation of CCA projections across trials:

$$\mu \in \mathbb{R}^{T_c \times k}, \quad \sigma \in \mathbb{R}^{T_c \times k}, \quad (19)$$

where μ represents the average representation of each condition in the canonical space, and σ represents the corresponding variability. We visualize the distribution of μ along the first few canonical components (CC1–CC3) to assess how well the representational geometry aligns with task-relevant variables.

If a common neural schema exists, CCA should reveal a stronger alignment of neural activity across problems. This implies that the Canonical Components (CCs) derived from the analysis of one set of problems should enable identification of trial types from the CCs derived from other problems. To test this prediction, we conducted a decoding analysis. As shown in Fig. S6a, our results demonstrate strong correlations between the CCs derived from the training and test data in cross-problem decoding. For instance, CC1 exhibits a correlation of 0.95, CC2 0.64, and CC3 0.77, which is consistent with experimental data, shown in Fig. S6b. Furthermore, similar to the common manifold observed in rat OFC activity space, the first three CCs are also interpretable: CC1 encodes the current value (reward status), CC2 captures odor overlap (distinguishing unique vs. shared odors), and CC3 reflects positional alternation (spatial transitions). Similar CCA results are also found across subjects, as shown in Fig. S6c,d. These findings further suggest that during task schema learning, our model develops a common schema, shared across problems and models under different seed conditions, and that this common schema possesses interpretable manifold dimensions akin to those found in experimental data.

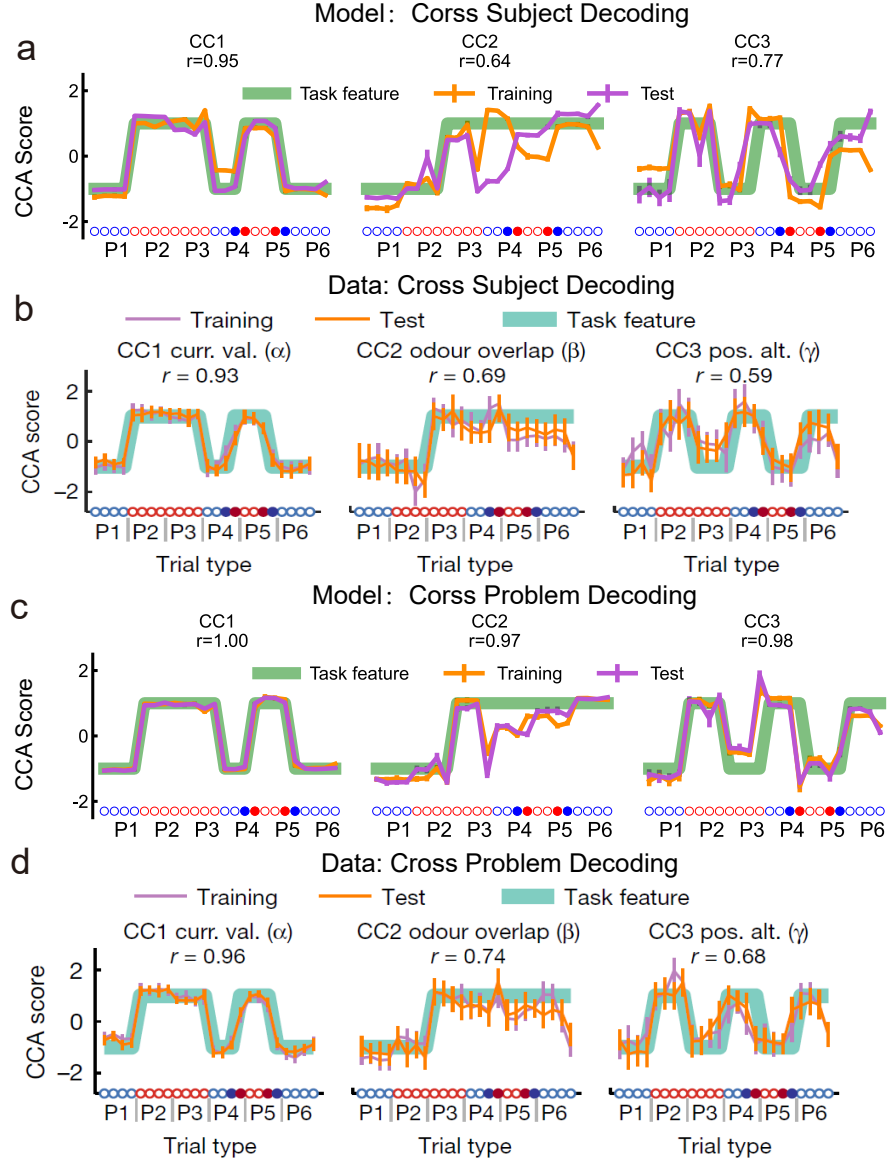


Figure S6: Canonical correlation analysis (CCA) of dimensionally reduced neural activity from the RNNs. We performed CCA on the PCA-reduced RNN activity, R^{PCA} , as described in Sec. C.1 of the Supplementary Text, and visualized the top three canonical components (CCs). The *train* and *test* represent two separate groups of subjects (or problems), and CCA is performed on each group independently. The mean and standard deviation of the results across 20 trials are calculated. The value r denotes the correlation between the canonical components obtained from the CCA of the training and testing groups. Blue and red dots represent rewarded and non-rewarded trials, respectively. Filled circles highlight key trial positions: S2a4+, S2b4-, S2a5-, and S2b5+. (a) The top three CCs reveal alignment of neural representations across differently seeded instances of the shaped RNN model. (b) The top three CCs reveal alignment of OFC neural representations across different rats (adapted from[1]). (c) The top three CCs reveal alignment of neural representations across different problems using the same shaped RNN model. (d) The top three CCs reveal alignment of OFC neural representations across different problems within the same rats (adapted from[1]).

C.5 Investigating Schema Code Development in RNNs under Extended Training Problems

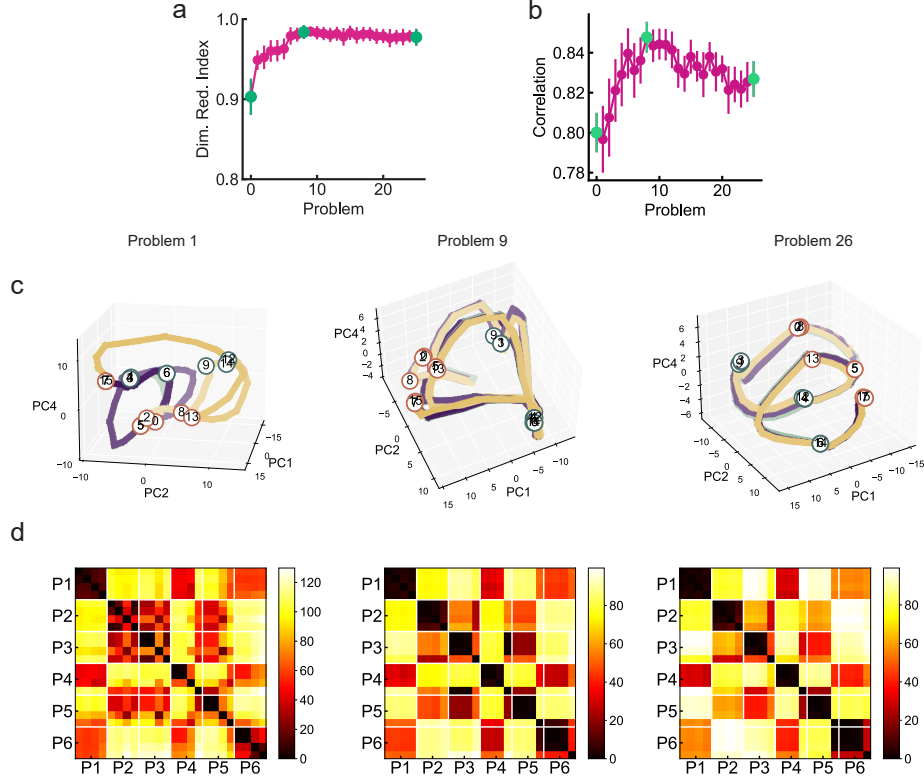


Figure S7: RNN dynamics and RDMs across extended problem sets. (a) Dimensionality compression over time in the RNN across 26 odor-sequence problems. (b) Pearson correlation between experimental and model RDMs across 26 odor-sequence problems. (c) PCA visualization of neural trajectories for S1a, S1b, and S2a in Problem 1, Problem 9 (highest correlation), and Problem 26. (d) RDMs for RNNs in Problem 1, Problem 9, and Problem 26. All data are averaged over 5 seeds.

Zhou et al.’s experiments demonstrated that rats exhibit learning-to-learn effects and develop a low-dimensional schema code[1]. Remarkably, these effects emerged after training on only five problems. This is particularly surprising given that even primates typically require exposure to hundreds of problems in simple sensorimotor tasks to display a convergent learning-to-learn behavior[4]. These observations raise two key questions: Has the schema code in the rat OFC already reached a well-developed and stable state? And if training were extended to more problems, how would the schema code further evolve?

To explore this discrepancy, we extend the training of our RNN model across a larger number of problems. Our goal is to investigate how the model’s neural dynamics evolve and whether the resulting task geometry mirrors the schema development observed in rats. As illustrated in Fig. S7a, the RNN’s coding space progressively becomes more low-dimensional with continued training, eventually reaching a plateau. This dimensionality reduction suggests a gradual abstraction of task structure across problems.

Moreover, we observe that the Pearson correlation between the model’s internal representations and experimental data first increases and then declines. This non-monotonic trend may reflect an initial alignment with empirical data, followed by a divergence as the model abstracts further beyond the rats’ current schema representations.

This abstraction process is further evidenced in Fig. S7c,d, where sequence attractors corresponding to tasks S1a, S1b, and S2a converge into a unified attractor. This convergence occurs because these tasks share an identical reward transition structure. Taken together, these findings suggest that if rats

were trained on more problems, their schema representations might continue to evolve, giving rise to a higher-level abstract manifold that encodes common structural elements across different sequences.

D Keyword Spotting Task: Model and Training Details

The keyword spotting task shares a structural analogy with the odor-sequence task - both require building complex sequences from simpler components (e.g., phonemes to words, and odor-rewards to odor sequences, respectively). However, the keyword spotting task introduces critical real-world variability (such as differences in speaker acoustics, timing, and pronunciation) that was absent in the initial task, thereby significantly increasing the generalization challenge. Despite this increased difficulty, we find that the model efficiently learns the structured schema, demonstrating our approach’s robustness and transferability. The detailed description of the keyword spotting task follows below.

D.1 Task Description

Task Primitive Learning. In this initial stage, the model is trained to recognize distinct phonemic units using speech data from a single speaker. The raw audio signals are processed with a sliding window of 25 milliseconds and a stride of 10 milliseconds, and then transformed into Mel-Frequency Cepstral Coefficients (MFCCs), which serve as input features to the model. To ensure feature stability and consistency across different samples, the MFCC vectors for each word are normalized along the temporal axis by subtracting the mean and dividing by the standard deviation. The data are from the TIMIT dataset [5].

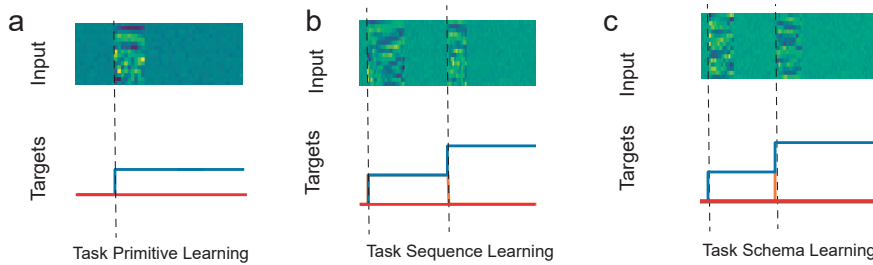


Figure S8: An example of input sequence vectors and corresponding hierarchical target functions for (a) Task Primitive Learning, (b) Task Sequence Learning, and (c) Task Schema Learning. In (a), the input consists of MFCC features with a single phoneme stimulus presented in a fixed-length trial; in (b), phonemes are arranged sequentially with variable intervals; in (c), similar sequences are drawn from multiple speakers to test generalization.

The training dataset consists of four words — “water”, “wash”, “had”, and “year” — each mapped to a specific sequence of phonemes: water corresponds to [“w-ao”, “dx-axr”]; wash to [“w-ao”, “sh”]; had to [“hv”, “eh-dcl”]; and year to [“y-ih”, “axr”]. To guide the model’s learning, each phoneme is assigned a target vector representing its identity as a one-hot vector, with the non-zero entry set to 2.

As illustrated in Fig. S8a, stimuli are presented randomly between time steps $t = 5$ and $t = 10$ within a total trial duration of 50 time steps. Upon the onset of the phoneme stimulus, the model is presented with the corresponding target vector, which persists until stimulus termination.

Task Sequence Learning. Building upon the phoneme attractor representations learned in the primitive phase, the task sequence learning phase trains the model to represent complete word-level phoneme sequences. Following the approach proposed in Zou et al.[6], the phoneme sequences are augmented by inserting background inputs between consecutive phonemes, as illustrated in Fig. S8b. These background intervals are randomly sampled to span between 5 and 25 time steps. This augmentation encourages the network to develop a hierarchical, tree-like attractor structure in its hidden state dynamics, promoting robust sequence learning across variable timing conditions[6]. As shown in Fig. S8b, a hierarchical ramping target function is applied in training. This function dictates that the activation associated with each phoneme target gradually rises and falls in accordance with the phoneme’s onset. This approach helps the model capture the temporal dependencies between

phonemes more effectively. It is important to note that, although the model is trained using temporally augmented sequences with inter-phoneme background input, during testing, the model is directly evaluated on non-augmented, unsegmented word inputs.

Task Schema Learning. In this phase, speech data from multiple speakers is introduced to improve the model’s generalization to speaker variability. As in the sequence learning phase, phoneme sequences are augmented by inserting background inputs with randomly sampled durations between 5 and 25 time steps (Fig. S8c). A hierarchical ramping target function is applied to each phoneme, guiding the network toward stable, structured attractor representations. Each word includes roughly 300 utterances from different speakers. Four utterances per word are used for training, while the rest are reserved for testing. This setup allows for a clear evaluation of the model’s ability to generalize across speakers.

To enhance robustness, additive Gaussian noise (zero mean, variance of 0.5) is applied throughout all training phases.

D.2 Model and Training Setup

The model architecture used here are LSTM and GRU. The input to the network consists of MFCC features with 16 dimensions. The recurrent layer comprises 200 hidden units, followed by an output layer of 4 dimensions corresponding to the target vector. All connection weights are initialized using the Glorot initialization method to facilitate efficient convergence.

During the task primitive learning phase, the model is trained over 10 epochs, with each epoch comprising 40 trials. Each phoneme appears 10 times per epoch, ensuring balanced exposure. A batch size of 4 is used, and the learning rate is set to 0.001. The optimization process updates all network weights using backpropagation through time.

For the task sequence learning and task schema learning phases, training is carried out over 50 epochs, with each epoch containing 20 trials. A batch size of 4 and a learning rate of 0.001 are used throughout this phase.

References

- [1] Zhou, J., Jia, C., Montesinos-Cartagena, M., Gardner, M. P., Zong, W., & Schoenbaum, G. (2021). Evolving schema representations in orbitofrontal ensembles during learning. *Nature*, 590(7847), 606-611.
- [2] Wilson, R. C., Takahashi, Y. K., Schoenbaum, G., & Niv, Y. (2014). Orbitofrontal cortex as a cognitive map of task space. *Neuron*, 81(2), 267-279.
- [3] Sussillo, D., & Barak, O. (2013). Opening the black box: low-dimensional dynamics in highdimensional recurrent neural networks. *Neural computation*, 25(3), 626-649.
- [4] Harlow, H. F. (1949). The formation of learning sets. *Psychological review*, 56(1), 51.
- [5] Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S. & Dahlgren, N. L. (1993). DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus CDROM NIST
- [6] Zou, X., Chu, Z., Guo, Q., Cheng, J., Ho, B., Wu, S., & Mi, Y. (2023). Learning and processing the ordinal information of temporal sequences in recurrent neural circuits. *Advances in Neural Information Processing Systems*, 36, 33999-34020.