FRONT MATTER

Title

- Full title: Beyond the Delay Neural Dynamics：a Decoding Strategy for Working Memory Error Reduction

Authors

Zeyuan Ye[1,2,3,4]†, Haoran Li[1]†, Liang Tian[1,5,6,7*], Changsong Zhou[1,2,5,8*]

Affiliations

[1]Department of Physics, Hong Kong Baptist University, Hong Kong, China
[2]Centre for Nonlinear Studies and Beijing-Hong Kong-Singapore Joint Centre for Nonlinear and Complex Systems (Hong Kong), Hong Kong Baptist University, Hong Kong, China
[3]Institute of Interdisciplinary Studies, Hong Kong Baptist University, Hong Kong, China
[4]Department of Physics, Washington University in St. Louis, St. Louis, Missouri, USA
[5]Institute of Computational and Theoretical Studies, Hong Kong Baptist University, Hong Kong, China
[6]Institute of Systems Medicine and Health Sciences, Hong Kong Baptist University, Hong Kong, China
[7]State Key Laboratory of Environmental and Biological Analysis, Hong Kong Baptist University, Hong Kong, China
[8]Life Science Imaging Centre, Hong Kong Baptist University, Hong Kong, China

† These authors contributed equally.
[*]Correspondence and requests for materials should be addressed to L.T. or C.Z. (email: liangtian@hkbu.edu.hk; cszhou@hkbu.edu.hk)

Abstract

Understanding how the brain preserves information despite intrinsic noise is a fundamental question in working memory. A typical delayed-response task consists of a delay epoch for maintaining information, go and response epochs (decoding phase) for decoding maintained neural state to an output action. While previous works have focused on the delay neural dynamics, as another key phase in working memory, the role of decoding phase in memory error reduction has not been investigated: what and how the maintained neural state is decoded to an output action; and how this decoding process can help reducing the memory error? We address these questions by training artificial recurrent neural networks (RNNs) to execute a color delayed-response task. We found the trained RNNs learned to reduce the memory error of the high-probability-occur colors (common colors) by decoding/attributing a broader range of neural state space to common colors. This decoding strategy can be further explained by both the converging neural dynamics and a non-dynamic, biased readout process during the decoding phase. Our findings provide testable prediction of the critical role of the decoding phase in memory processes, suggesting that neural systems deploy multiple strategies across different phases to reduce the memory errors.

Significance Statement

Preserving information under noise is of crucial in working memory. A typical delayed-response experiment consists of a delay epoch for maintaining information, and a go and response epoch (decoding phase) for decoding the maintained neural state into output information. While the delay neural dynamics has been intensively studied, the impact of the decoding phase on memory error reduction remains unexplored. We trained recurrent neural networks (RNNs) on a color delayed-response task and found that RNNs reduce memory error of a color by decoding a larger portion of neural state to that color. This strategy is partially supported by a non-dynamic readout process. Our results suggest that neural networks can utilize diverse strategies, beyond delay neural dynamics, to reduce memory errors.

MAIN TEXT

Introduction

Working memory is the ability to maintain information for a short period of time without external stimuli. However, this ability is not perfect. Due to the noise, without additional error-correcting mechanisms, the neural population activity will diffuse away from its original state, leading to a large memory error[1–4]. The neural mechanisms utilized by the brain to mitigate these memory errors remain unclear.

A typical delayed-response task contains a perception epoch for sensing the stimuli, a delay epoch for maintaining the stimuli, and finally, a go and a response epoch for initiating an action that indicates the memorized information's value. Previous works have mainly focused on the neural dynamics during the delay epoch[1–11]. For instance, the neural system can form discretized attractors to represent some information values[6–9]. An attractor is a special neural population state; any small deviation from the attractor state will be brought back. This unique property of attractors can stabilize neural population states against random deviations due to noise, thereby reducing memory errors. This attractor-based memory error correcting mechanism has been supported by experimental behavior data[7], experimental neural recordings[12], and simulations from artificial neural networks [3,9,13]. It can also be implemented by artificial neural networks using biologically plausible synaptic rules [6,14].

However, despite also being key epochs, the role of the go and response epochs has largely been neglected. From the information processing perspective, the go and response epochs act as a decoding mapping, which maps the maintained delayed neural state to an output action[15–20]. A change in the decoding mapping will lead to a significant change of behavioral performance. This importance of decoding can also be demonstrated in the Brain-Computer Interface (BCI) experiments[20,21], where the BCI functions as a decoder, mapping neural population state to external actions (e.g. cursor movement). Alterations in the BCI's decoding mapping would lead to significant action errors, necessitating relearning of the new decoding pattern by the animal. Similarly, in working memory, the delay neural population state must progress through the go and response epochs, be decoded to muscles signals for an output action (e.g. saccade).

More precisely, this paper concerns about the decoding mapping by the go and response epochs. What is the decoding mapping from a maintained neural state at the end of the delay to an output information? Whether and how such mapping helps reducing the memory error? What processes occur during the go and response epochs to establish this decoding mapping? Addressing these

questions is important for understanding the diverse strategies, beyond just delay neural dynamics, of the neural system to reduce the memory error.

In this paper, we trained artificial recurrent neural networks (RNNs) to perform the color delayed-response task, where the color in each trial was sampled from a prior distribution with a few high-probability colors (common colors)[9,22,23]. We found that the trained RNN exhibited less memory error on common colors, which aligns with previous behavioral experiments[7]. This provided an opportunity to study what neural mechanisms the RNN adopted to reduce memory errors on common colors. We discovered two primary mechanisms. First, in agreement with previous dynamic theory[8], the neural system formed attractors to encode common colors in the maintenance phase. Second, in the decoding phase, an extended portion of neural population states was mapped (in other words, decoded) to common colors, which allows stronger tolerance to noise. This noise-robust decoding mapping can be further explained by the converging dynamics during the go cue and response epoch, and also the biased readout from recurrent neural state to common colors. We further proposed an approximation theory which naturally decompose the memory error into dynamic and decoding components – neglecting decoding component will lead to a failure of explaining the RNN's memory error. Our results emphasize the importance of the decoding phase and propose experimentally testable predictions.

Results

RNN architecture and training

The vanilla RNN architecture in our study includes 12 perception neurons, one go neuron, 256 recurrent neurons, and 12 response neurons (Figure 1A and Methods). The recurrent neurons are entangled with a slight amount of noise. In each trial, a color is sampled from an environmental prior distribution which contains four peaks (mirroring the behavioral experiment[7]) indicating four common colors that are more likely to occur. This input color, sampled from the environmental prior, is detected by the perception neurons through predefined tuning curves (von Mises functions with shifting mean positions, see Figure 1A) and is subjected to additional minor Gaussian noise perturbations (detailed in Methods). The activity from the perception neurons is then fed forward to the 256 recurrent neurons. During the delay period (with its length randomly determined from a uniform distribution in each trial), the RNN is required to use these recurrent neurons to preserve the color information, even in the absence of input signals. Following the delay, the go neuron receives a signal, initiating the response action. During the response epoch, the response neurons become active. These neurons, equal in number to the perception neurons, are expected to generate activity mirroring that of the perception neurons during the perception epoch (but without the noise, see Methods). The activity of these response neurons is then temporally averaged and translated back into color information using the population vector method (Equation 6). Overall, the trial can be viewed as comprising three phases of information processing: encoding (perception), maintenance (delay), and decoding (go and response).

Each RNN underwent progressive training across multiple trials[24] (Figure 1B and Methods). The training process was divided into two critical stages. The initial stage, termed 'pretraining', involved training the untrained RNN on several trials using colors sampled from a uniform prior environment. Following this, the second stage involved 'retraining' the pretrained RNN in a biased prior environment, resulting in what we term the 'Biased RNN'. For baseline comparison, another set of pretrained RNNs was retrained using a uniform environment, and these are referred to as 'Uniform RNNs'. Behavioral analysis demonstrated that the trained RNNs successfully replicated multiple behavioral phenomena observed in previous experimental studies[7,25], including increased memory errors with longer delays, a biased output color distribution, and a propensity to output common colors (SI Figure 1).

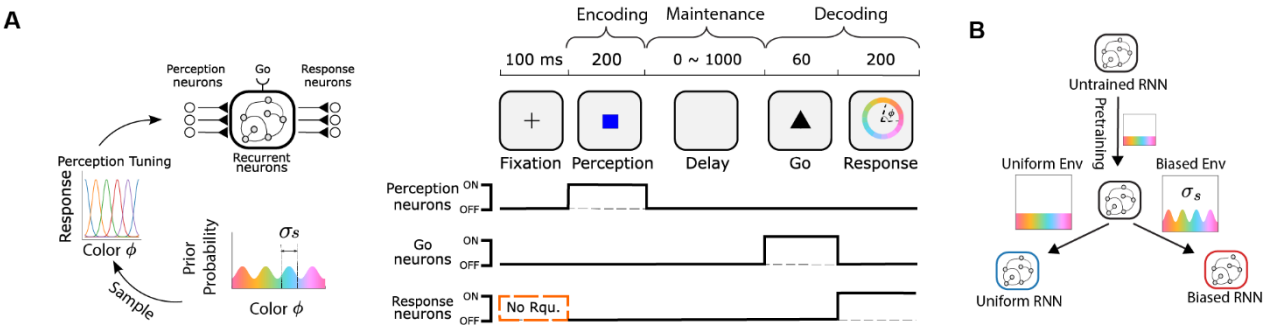


Fig 1. RNN architecture and training. (**A**) An environment is defined as a probability prior function for the input color. In the perception epoch of each trial, a color is sampled from the prior, and then perceived by the RNN using perception neurons with predesigned tuning curves. Following the perception epoch, the RNN goes through a delay epoch with a length uniformly sampled from 0 to 1000 ms. Next, a go signal is perceived by the go neuron. Finally, the RNN needs to use response neurons to recall the perceived color, by activating response neurons the same way as perception neurons during the perception epoch. Except regularization terms, loss function is only applied on the response neurons, which has no requirement in fixation epoch (No Rqu.) and should be silent in the perception, delay and go epochs. (**B**) An RNN is trained by multiple trials progressively. An untrained RNN was pretrained in a uniform environment, then retrained in a biased environment ($\sigma_s = 12.5°$ as an example). As a baseline comparison, pretrained RNN was also retrained in a uniform environment with the same number of trials. Overall, this progressive training scheme resulted in the trained Biased RNNs and Uniform RNNs.

Cross-decoding experiments show the decoding phase is crucial in memory error reduction.

We assessed the memory errors in both the Biased and Uniform RNNs. It's important to note that in this study, color is treated as a circular variable ranging from (0°to 360°). Therefore, the memory error in a trial is defined as the circular subtraction of the RNN's output color from its input color. The overall memory error across multiple trials is determined by the root-mean-squared error of these individual trial errors.

In our experiment, the memory errors of the Uniform and Biased RNNs were measured in an identical biased environment ($\sigma_s = 12.5°$, which is the same environment the Biased RNNs were trained in, over 5000 trials for each RNN). As expected, the Biased RNN exhibited smaller memory errors in this environment, given its training specificity (Figure 2A, $p < 10^{-3}$, Wilcoxon signed-rank test). Mathematically, the memory error is the weighted average (weighted by the prior) of the memory error for every fixed input color. Hence, we also examined the memory error for each fixed input color (Figure 2B). For the Uniform RNN, memory errors were uniform across all input colors. In contrast, the Biased RNN showed a reduced memory error for the common color (at 130°) compared to other input colors and also to the same input color in the Uniform RNN.

This observation of reduced memory error in the Biased RNN, especially for common colors, aligns with findings in previously modeled networks[6] and experiments[7]. It presents an opportunity to delve into the neural mechanisms the Biased RNN employs to achieve this reduction in memory error compared to the Uniform RNN.

Prior theoretical studies have primarily focused on delay dynamic mechanisms for minimizing memory errors. Building upon this, our study hypothesizes the crucial role of the decoding process

as well. We explored this hypothesis through a cross-decoding experiment (Methods), wherein the non-decoding (fixation, perception, and delay epochs) and decoding phases (go and response epochs) were separated (See Figure 2C, upper panel). In this experiment, one RNN model (model 1) was selected at random from either the Biased or Uniform groups. This model underwent a trial with a fixed common color input, and the recurrent neural state at the delay's end was recorded. This state was then used to initialize a second RNN model (model 2), randomly chosen from the other category (Biased or Uniform). Model 2 proceeded directly to the decoding phase and output a color. After repeating this process across 500 trials, we computed the memory error for each pair of RNNs. This entire process was conducted for 50 random pairs for each category combination (Biased & Uniform, Biased & Biased, Uniform & Uniform, Uniform & Biased). Our results show that combinations involving a biased decoding phase (where model 2 is a Biased RNN) consistently demonstrated statistically significant reduced memory errors for the common color (Figure 2C) compared to those involving a uniform decoding phase (where model 2 is a Uniform RNN) (Bias & Bias vs. Bias & Uniform; Uniform & Bias vs. Uniform & Uniform). This provides direct evidence that the trained Biased RNN developed mechanisms during the decoding phase to decrease memory error for common colors.

Furthermore, our findings indicate that even when the decoding phase category is consistent, variations in the prepared delay state categories lead to differing memory errors (Figure 2C, see Bias & Bias vs. Uniform & Bias comparisons). This observation underscores the significance of the dynamics prior to the decoding phase in the reduction of memory errors.

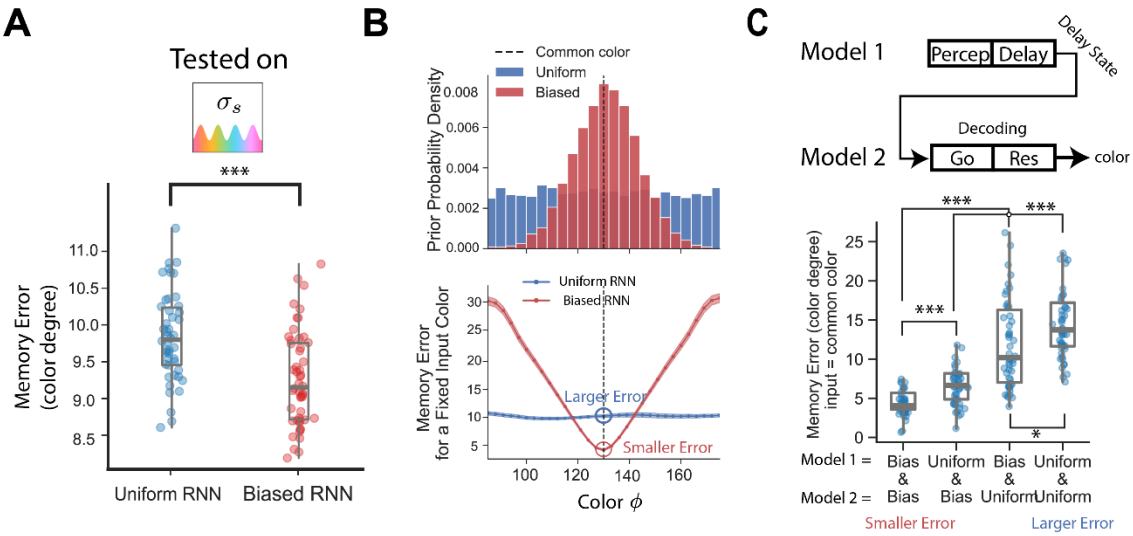


Figure 2. Cross-decoding experiments demonstrate the critical role of the decoding phase in reducing memory errors for common colors (common colors are fixed at $40^\circ, 130^\circ, 220^\circ$, and 310°). Delay length is fixed to 800 ms. **(A)** Uniform and Biased RNNs, with 50 of each type, underwent testing in a biased environment ($\sigma_s = 12.5^\circ$). Each RNN was subjected to 5000 trials, and the memory error for each RNN (shown as individual dots) is calculated as the root-mean-square deviation across these trials. The Biased RNNs exhibited significantly smaller memory (***: $p < 10^{-3}$ Wilcoxon signed-rank test). **(B)** Top: The prior distribution of colors around a common color (130°, black dashed line). Bottom: Memory error for various input colors, assessed over 5000 trials for each input color and RNN. The colored lines represent the mean memory errors across the 50 RNNs, with error bands indicating standard errors. Outlier RNNs are removed. **(C)** Cross decoding suggesting the importance of decoding phase in reducing memory error (e.g., Bias & Bias vs. Bias & Uniform). Delay state is prepared from a model 1, then be decoded by model 2. Each dot is a sampling of one model 1 and one model 2. 50 samplings for each model1-model2 category combination (Bias or/and Uniform). ***: $p < 10^{-3}$; *: $p < 5 \times 10^{-2}$; Wilcoxon signed-rank test. Boxes indicate the interquartile range between the first and third quartiles with the central mark inside each box

indicating the median. Whiskers extend to the lowest and highest values within 1.5 times the interquartile range. Outlines are not shown.

A hypothesis of the memory error reduction by both dynamic and decoding mechanisms

The cross-decoding experiments conducted underscore the pivotal roles of both the delay and decoding phases (go cue and response epochs) in influencing memory errors. Previous hypotheses have suggested that reduced neural dispersion dynamics during the delay might contribute to error diminution[7]. However, our study introduces a novel hypothesis concerning the importance of the decoding phase in error reduction. Conceptually, the state of a neural population can be represented as a point within a multidimensional state space, where each axis represents the activity of an individual neuron (Figure 3). The decoding process, in this framework, involves mapping distinct portions of this state space to specific colors. This mapping is geometrically equivalent to assigning state occupancy to each color. Imagine a scenario where neural states, despite being noisy and dispersed, fall within a region of the state space mapped to a single color; the output color remains constant despite the dispersion of neural states. Thus, colors with larger state space occupancies exhibit greater tolerance to noise, subsequently diminishing memory errors. In summary, two primary mechanisms are proposed for reducing memory errors (Figure 3): (1) reducing the dynamic dispersion of neural population states during the delay, and (2) assigning a wider range of neural states to common colors, a process we term 'larger state space occupancy by a color'. This paper focuses on systematically exploring the latter mechanism, offering new insights into the neural underpinnings of memory error reduction.

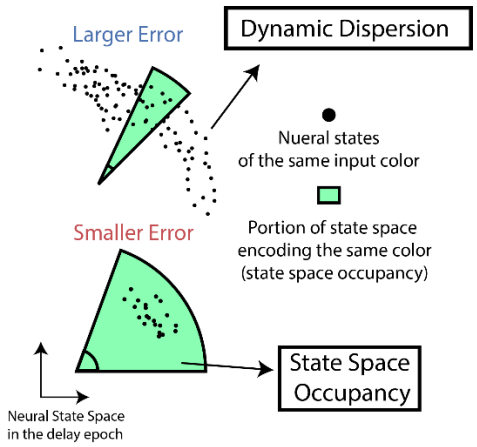


Figure 3. A hypothesis decomposing the memory error reduction mechanism into a dynamic part (dynamic dispersion) and decoding part (state space occupancy). Illustration of the hypothesis: (1) Each dot is one neural population state at the end of the delay, axes are single neural activity. Memory error can be reduced if the neural states are more stable (smaller dynamic dispersion). (2) Green region represents portion of state space that will be decoded into a color. Larger state space occupancy allocating to one color allows better tolerance of neural state dispersion, hence smaller error for that color.

RNN's neural activity is low-dimensional.

To test this memory error reduction hypothesis, we analyzed the neural population activity within the RNNs. We conducted 1000 trials for each RNN, with color inputs uniformly sampled, and with the RNN's noise component deactivated. The neural activity collected during these trials was then projected onto the first few principal components using principal component analysis (PCA, see

Figure 4A). The cumulative variance explained ratio from this analysis indicates that the neural population activity is essentially low-dimensional: requiring only 3 dimensions to account for over 90% of the variance in neural activity across the entire trial, and only 2 dimensions when focusing exclusively on the delay epoch.

The inherent low dimensionality of the neural activity in our study allowed for direct visualization by projecting the activity onto the first three principal components (PCs), as shown in Figure 4B. During the perception epoch, the neural states are influenced by the input colors, moving towards different directions within a ring-shaped manifold. In the subsequent maintenance phase, the delay epoch, these neural states remain relatively stable, although slight lateral movements are observed. Finally, in the decoding phase, the neural states are influenced by the go input during the go epoch, shifting along the third principal component. This is followed by a period of stability in a ring-like structure during the response epoch. These observations from the various phases of neural activity illustrate the dynamic yet structured nature of neural state transitions in different phases of task execution.

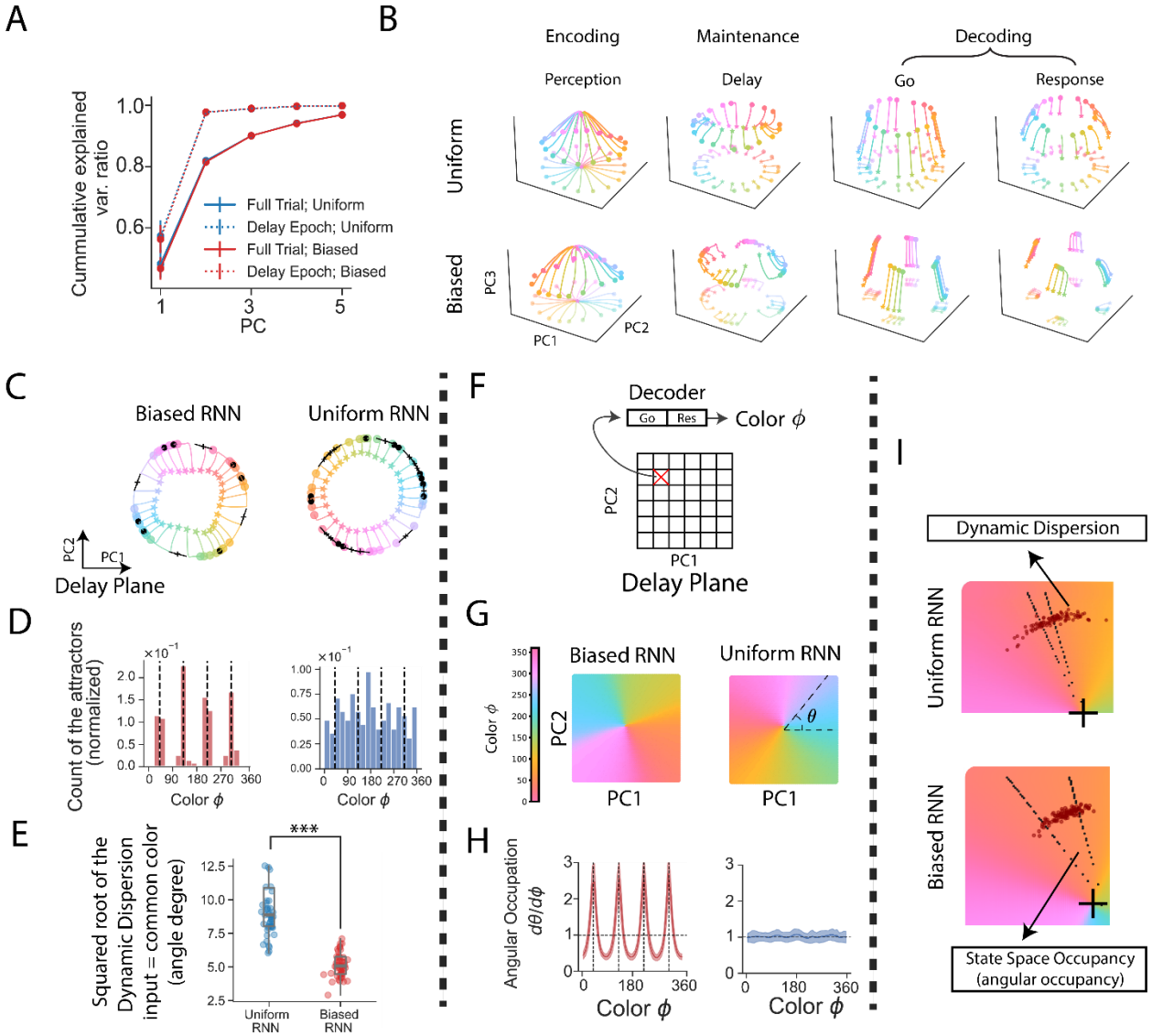


Figure 4. Reverse engineering of the RNNs shows evidence for the hypothesized memory reduction mechanisms (smaller dynamic dispersion and larger state space occupancy). (A) Dimensionality of the neural activity (1000 trials for each RNN). Dots are the mean across 50 RNNs, error bar is the standard deviation. Uniform: Uniform RNN; Biased: Biased RNN. Blue and red traces are almost overlap. **(B)** PCA

visualization of neural states. Stars denote the beginning of the epochs; dots represent the end. Each trajectory is one trial with color indicating the trial input color. **(C)** 2D visualization of two example RNN's neural states in the delay epoch. Black dots are the attractor points. Crosses are saddle points whose long bars indicating the unstable (positive eigenvalue) direction. **(D)** Distribution of attractors. More attractors were formed near common colors (dash lines) in the trained Biased RNN. Results concatenated attractors from 50 RNNs. **(E)** Neural states during the delay have smaller dynamic dispersion (see Methods) in Biased RNN when the input color is fixed to common color. Each dot is one RNN. ***: $p < 0.001$ Wilcoxon signed-rank test. Boxes indicate the interquartile range between the first and third quartiles with the central mark inside each box indicating the median. Whiskers extend to the lowest and highest values within 1.5 times the interquartile range. Outliers are not shown. **(F)** Decoding the delay plane by selecting mesh points and running through go and response epochs. **(G)** Decoding results of two example RNNs. Color is the decoded color. θ represents angle on the PC1-PC2 plane. **(H)** Angular occupancy--the size of angle on the delay plane used for encoding one unit of color. Solid lines are the mean across 50 RNNs, and error band is the standard error. (**I**) Two example RNNs illustrate the joint effect of dynamic dispersion and angular occupancy. Each black dot is the position of neural state (on the delay plane) at the end of delay of one trial (delay length = 800 ms), where trial's input color is fixed to a common color (40°). Dash line encloses the angular space for encoding 35° to 45°. Black crosses are the center of the PC1-PC2 plane. For Biased RNNs presented here, $\sigma_s = 12.5°$. RNN's intrinsic noise was turned off for better visualization, except in (E, I).

Biased RNN formed attractors near common colors to reduce the dynamic dispersion.

To evaluate whether dynamic dispersion during the delay phase is a key factor in reducing working memory errors, we firstly focused solely on the delay neural activity. Instead of projecting onto three dimensions, we projected this activity onto the first two principal components (PCs), in line with the suggestion in Figure 4A that two dimensions sufficiently describe the delay neural activity. We refer to this two-dimensional representation as the 'delay plane'. On this delay plane, neural states for different input colors formed a ring-like structure (Figure 4C).

Observations revealed minor motions along the tangent of this ring. This slight movement might be associated with fixed points on the ring. Fixed points can be categorized as attractors, which draw nearby neural states, or saddle points, which repel nearby neural states along the direction of positive eigenvalues. We computed the locations of these attractors and saddle points (Methods). In our Biased RNN example, attractors predominantly appeared in four positions (as seen in Figure 4C). These attractors also had more negative eigenvalues (indicating stronger attractive force) in more Biased RNNs (with smaller prior σ_s, SI Figure 2).

The positioning of the four attractors on the delay plane potentially aligns with the four common colors in the biased environment's prior probability distribution. To explore this possibility, we developed a RNN decoder. This decoder is tailored to translate neural states on the delay plane into their corresponding output colors. A RNN decoder is a faithful replication of the original RNN (Methods). To perform the decoding, we initialize the recurrent state of the decoder to match the given neural state. Subsequently, the decoder processes this state through the go and response epochs. The color decoded at the end of this process is identified as the output color.

We decoded the attractors of 50 Uniform RNNs and 50 Biased RNNs. The analysis revealed that in Biased RNNs, attractors were primarily mapped to common colors (Figure 4D), suggesting that dynamic dispersion near common colors is reduced in Biased RNNs. To validate this, we directly measured dynamic dispersion as follows: We conducted 1000 trials for each RNN, keeping the input color fixed at the common color and the delay length at 800 ms. Neural states at the end of the delay were collected, and their angles on the delay plane were computed. The variance of these

neural state angles represents the dynamic dispersion. The results (Figure 4E) indicate that the dynamic dispersion for the common color in Biased RNNs is smaller than in Uniform RNNs, supporting the hypothesis that RNNs reduce memory error by diminishing dynamic dispersion[8].

Biased RNN allocates larger angular occupancy to common colors

Besides dynamics, we also inspected the decoding strategy of the RNN. Decoding is defined as a mapping from the neural state at the end of delay epoch to an output color. The decoding mapping is entirely determined by the RNN decoder (Figure 4F), which decodes a neural state by continuing to run the neural state through go and response epochs, averaging response neurons activities across time (see Methods), finally mapping to output color by populational vector method (Equation 7). We used the RNN decoder to decode the delay plane (not the neural state from any particular trials). Example RNNs' results suggested that color information is represented as angles in the delay plane (Figure 4G).

Next, we investigated the quantitative relationship between angle and color. We sampled and decoded dense neural states along a ring (see Methods). This provided a numerical relation between the angle to the decoded color. The numerical differentiation of the angle by the color is called angular occupancy. Angular occupancy measures the size of angular space mapped to a unit change of color. Results indicate that, in the Biased RNN, common color has larger angular occupancy (Figure 4H). As a baseline comparison, the angular occupancy of the Uniform RNN is uniform. These results suggested that Biased RNN reduces the memory error (for common color) by allocating larger angular space.

We used two example RNNs to provide the intuition of how the dynamic and decoding strategy jointly contribute to the memory error reduction (Figure 4I). We run each of the RNNs through 1000 trials with fixed common color as input color (delay length fixed at 800 ms). Neural states at the end of delay were collected, and visualized on the delay plane. Besides, we also used two dash lines to enclose the portion of space encoding color $35°$ to $45°$, around the common color $40°$. It can be seen that, compared to Uniform RNN, Biased RNN has larger angular occupancy and smaller dynamic dispersion, confirming the hypothetic picture in Figure 3.

An approximate theory quantitatively relates the dynamic dispersion and angular occupancy to the memory error.

Figure 4I provides a visualization of how the dynamic and decoding strategy jointly contribute to the memory error reduction on common colors. Here we also provide an approximation formula showing the quantitative relation between the dynamic dispersion, angular occupancy and the memory error at the common color.

The central idea is the Taylor expansion (see Methods). Color ϕ is a function of angle θ in PC1-PC2 space. Denoting common color as ϕ_c, the corresponding angle as θ_c. Assuming that, when trial input is the common color, the actual angle of neural state θ does not deviate from the true color angle θ_c too much. Therefore, we expanded the color function as $\phi(\theta) \approx \phi_c + \left(d\phi/d\theta|_{\phi_c}\right)(\theta - \theta_c)$, where $d\phi/d\theta|_{\phi_c}$ is the reciprocal of angular occupancy. Inserting this approximation into the squared memory error $\epsilon^2(\phi_c) = \sum_{i=1}^{N}(\phi_i - \phi_c)^2/N$ (where ϕ_i is the output color in trial i), we can then obtain an approximation formula (Equation 1 and Figure 5A)

describing the memory error using angular occupancy ($d\phi/d\theta|_{\phi_c}$), dynamic dispersion ($\sum_{i=1}^{N}(\theta_i - \bar{\theta})^2 / N$) and a mean bias correction ($(\bar{\theta} - \theta_c)^2$)

$$\epsilon^2(\phi_c) = \left(\frac{d\phi}{d\theta}|_{\phi_c}\right)^2 \sum_{i=1}^{N} \frac{(\theta_i - \bar{\theta})^2}{N} + \left(\frac{d\phi}{d\theta}|_{\phi_c}\right)^2 (\bar{\theta} - \theta_c)^2, \qquad (1)$$

where the mean bias correction described the correction due to the shift of trial averaged mean angle $\bar{\theta}$ to θ_c.

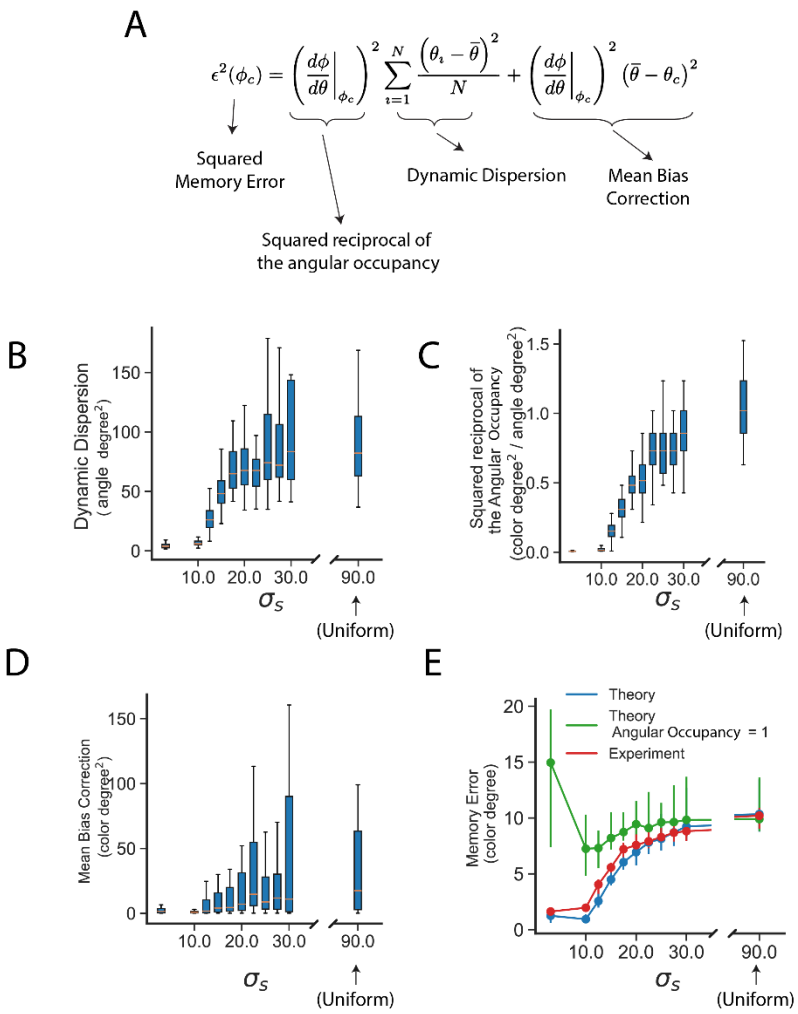


Figure 5. An approximation formula decomposes the memory error of a common color into a dynamic dispersion, angular occupancy, and correction terms. (A) An approximation formula for the squared memory error (see text). **(B, C, D)** Dynamic dispersions, squared reciprocal of the angular occupancy and mean bias correction for biased RNNs trained on different biased environment (50 RNNs for each σ_s). Smaller σ_s indicating narrower prior (see Figure 1A). Boxes indicate the interquartile range between the first and third quartiles cross 50 RNNs with the central mark inside each box indicating the median. Whiskers extend to the lowest and highest values within 1.5 times the interquartile range (see Methods). **(E)** Comparing the theory to the experimental RNN's memory error. Dots are the median memory error, and error bars show first and the third quantiles across 50 RNNs for each σ_s. Theoretical prediction was computed as (A). Experimental memory error of each RNN was measured by computing the averaged memory error of 5000 trials (fixing input at common color, outlier trial errors were removed). As a comparison, we also computed the prediction of theory but assuming trivial angular occupancy (equals to 1).

To test this approximation theory (Figure 5A), we trained biased RNNs on different environmental priors (50 RNNs for each prior) by changing the value of σ_s to control the prior width. The smaller

the σ_s, the narrower the prior, and the higher the prior probability of sampling a common color. We then computed dynamic dispersion, squared reciprocal of the angular occupancy and mean biased correction separately (see Methods). On the other side, we also computed the experimental memory error by running each RNN 5000 trials, fixing input color as common color.

We found the higher common color prior probability is (smaller σ_s), the smaller dynamic dispersion and the larger angular occupancy (Figure 5B, C) are. In general, the theoretical prediction has good alignment with the actual experimental memory error. To emphasize the importance of angular occupancy, we also computed theoretical prediction (Figure 5A) but setting angular occupancy to 1. We showed that neglecting angular occupancy factor (decoding strategy) will lead to a failure of explaining the RNN's memory error for small σ_s (Figure 5E).

Higher intrinsic noise leads to larger angular occupancy for common colors

This approximate theory is general in principle. It not only works for different environment priors, but in principle also works for the same environmental prior with different RNN's noise strength (Figure 6). The intuition is that, when the RNN's noise is larger, the information will become harder to maintain during the delay. As a tradeoff, the RNN will form stronger memory error reduction mechanism for those common colors. As an extreme example, if the noise is too large that the RNN can remember nothing, the optimal output color for the RNN are common colors since they have high priors. In this case, memory error reduction mechanism on common colors is very strong. Specifically, attractor would be more likely to accumulate on common colors and the angular occupancy for the common colors would be larger.

To test whether the RNN learned stronger error reduction mechanisms, we trained two RNNs with low-intrinsic noise (low-noise RNN) and high-intrinsic noise (high-noise RNN), respectively. It is found that high-noise RNN forms larger angular occupancy for common colors (Figure 6A, B), and is more likely to form attractors to encode common color (Figure 6C, D). Overall, RNN trained on larger noise environment has larger angular occupancy (Figure 6F). The theoretical prediction based on the dynamic dispersion, space occupancy and mean bias correction (Figure 6E, F, and G) also aligns with experiment memory error well, and better than theory without considering the decoding strategy (angular occupancy = 1) (Figure 6H). Note the error studied here is only the error of the common colors. It can be observed that total error across different colors naturally increases under larger noise (Figure 6I).

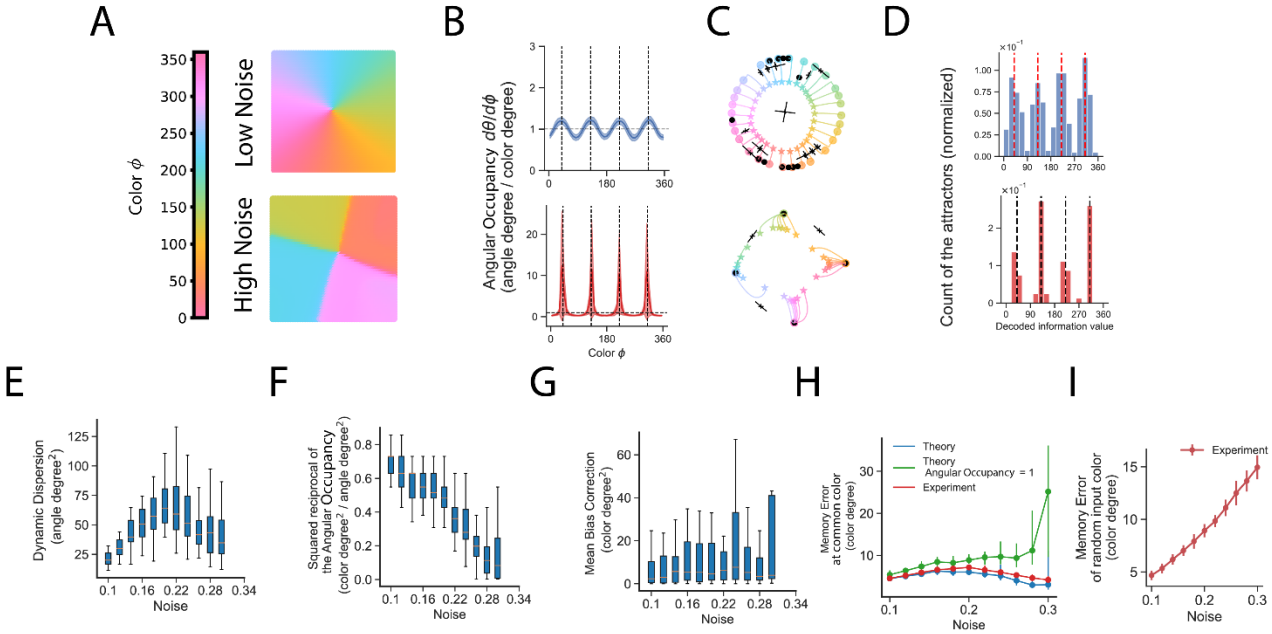


Figure 6. The results for different noise strength also support that angular occupancy is an important factor to reduce memory error at common color. (**A**) Decoding the delay plane for two example RNNs trained on low-noise (noise strength σ_{rec} = 0.1) and high noise (σ_{rec} = 0.3). (**B**) Angular occupancy. Dash vertical lines are common colors, Solid blue/red line indicates the mean of 50 RNNs and error bands are standard deviation. (**C**) Neural dynamics during delay. Each trajectory is the delay of one trial with color indicating the input color. Stars are the beginning of delay; dots are the ends. Black dots are attractors, black crosses are saddles with long bar indicating the positive eigenvalue directions. (**D**) Number of attractors for different color. Attractors were concatenated from 50 RNNs. (**E, F, G**) Dynamic dispersion, squared reciprocal of the angular occupancy and mean bias correction measured for 50 RNNs for each of the noise strength (see Methods). Boxes indicate the interquartile range between the first and third quartiles cross 50 RNNs with the central mark inside each box indicating the median. Whiskers extend to the lowest and highest values within 1.5 times the interquartile range. (**H**) Comparing experimental results with theoretical predictions for memory error. Dots are the median across 50 RNNs and error bar shows first and third quantiles. In the experimental part, memory error of each RNN is the average of 5000 trials error fixing input as common color (outlier trial error was removed). (**I**) Experimental error same as (H), but input color was randomly sampled from an environmental prior $\sigma_s = 17.5°$ instead of fixing to a common color. All RNNs were trained on environmental prior $\sigma_s = 17.5°$. Noise was turned off after training in computing panel (A, B, C, D, F).

Larger angular occupancy is due to both dynamics and biased readout matrix in the decoding process.

We have shown that the decoding strategy (i.e., angular occupancy to a color) is an important mechanism for reducing memory errors. However, it is unclear what happens inside the decoding process that can lead to such a non-uniform angular occupancy. We study this problem by dividing and conquering. The decoding process can be roughly separated into three stages (Figure 4B, 7A). (1) Go Dynamics: a go cue signal pushes the recurrent neural states from the delay plane to the response plane. (2) Response Dynamics: recurrent neural states can dynamically move during the response epoch (epoch length = 200 ms). (3) Readout: during the response epoch, recurrent neural states were readout (Equation 3 in Methods) into response neural activity, then mapped into output colors (Equation 7). This readout does not involve dynamics, it is simply multiplying the recurrent

neural states with a readout matrix with an additional bias current constant, then transforming into output color. Biologically speaking, readout matrix may involve the neural connections from motor cortex to muscles (for actions); bias current constant reflects the intrinsic property of each neuron (e.g. describing the diversity of neural firing threshold). We inspect possible mechanisms leading to non-uniform angular occupancy for each of these three stages (Go Dynamics, Response Dynamics, and Readout) separately.

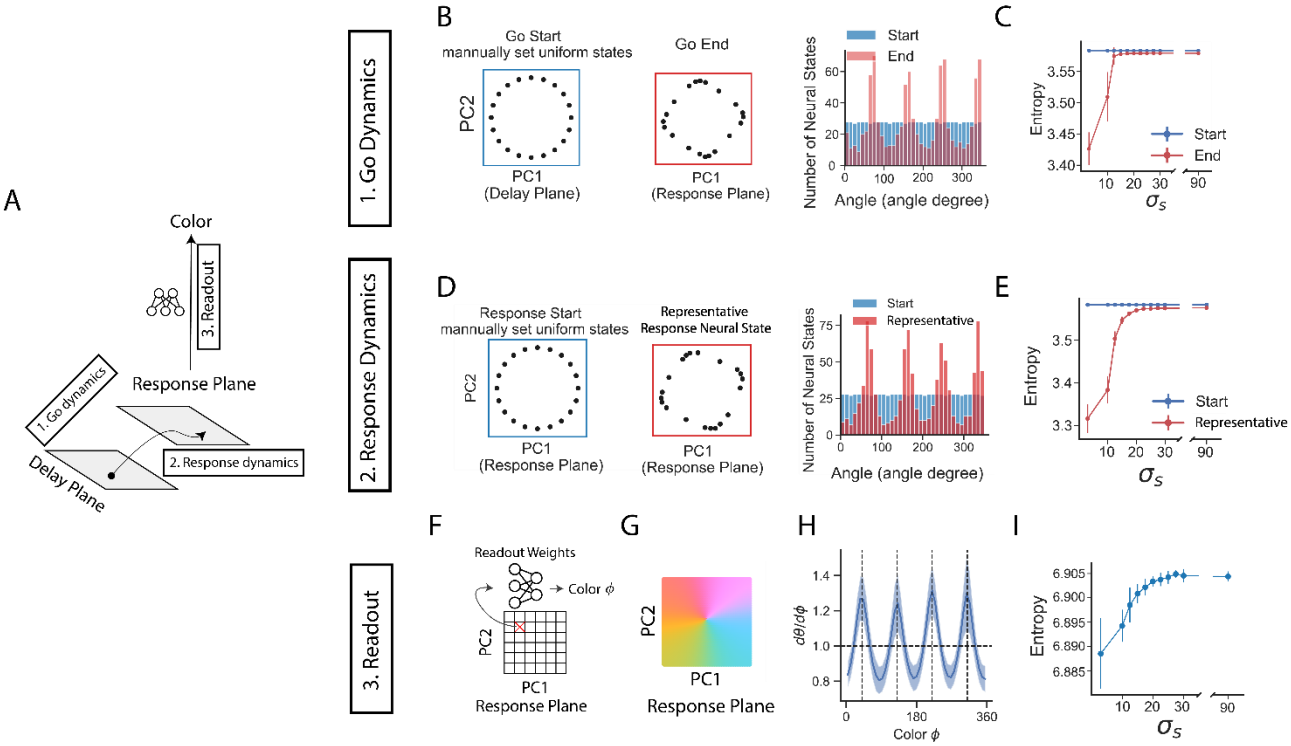


Figure 7. Larger angular occupancy (for common color) is due to both dynamics and readout mechanisms during the decoding process. The single example RNN shown in this figure was trained on prior $\sigma_s = 3°$; delay epoch length is fixed to 800 ms. **(A)** The decoding process roughly contains three stages (see Figure 4A, B): (1) Go cue drives neural states from delay plane to response plane; (2) Neural dynamics during the response epoch (200 ms); (3) Readout from recurrent neural states to output color. **(B)** We isolated the effect of Go Dynamics, by firstly fixing neural states uniformly along a ring in the delay plane, then letting the neural states evolve through go epoch. Left two boxes show one example RNN's neural states at the start and end of Go epoch. Right: the distribution of 1000 neural states of the same example RNN. **(C)** Entropy measures how non-uniform the neural states distribution is. Dots are the mean entropy across 50 RNNs (with outlier RNNs removed), error bar is the standard deviation. **(D, E)** Same as B, C, but the neural states were initialized in the response plane and evolved through response epoch. Instead of showing the neural state at the end of the response epoch, here we showed the neural states averaged across time in response epoch (representative response neural states) since which are more relevant for decoding color (see Equation 7). **(F)** Decoding response plane, by selecting mesh points and then readout into color through a readout matrix and bias currents (see Methods, Equation 3 and 7). No dynamics is involved. **(G)** The example RNN's decoded response planes. **(H)** Angular occupancy in the response plane. Line shows the mean of 50 RNNs ($\sigma_s = 3°$), and error band is the standard deviation. Vertical dash lines: four common colors. **(I)** Entropy of angular occupancy, dots are the mean and error bar are standard deviation (outlier RNNs removed).

In the Go Dynamics, tangent motion is a possible mechanism leading to larger angular occupancy for common color. We isolated the go epoch and measured its dynamics. At the beginning, we sampled a uniform ring of neural states (Figure 7B) in the delay plane, then let all these states evolve

through go epoch. At the end of the go epoch, we collected those neural states and fitted PC1-PC2 plane again (response plane). We found that neural states accumulated to a few positions. The degree of accumulation can be quantified by measuring the entropy of neural state distribution. The smaller the entropy is, the more likely the neural states tend to accumulate to a few positions. Figure 7C suggests that neural states accumulate happens when the environment prior σ_s is small. This suggested that non-uniform motion during the go state is a possible reason leading to a non-uniform angular occupancy in the delay plane.

Similarly, non-uniform motion during response epoch may also contribute to non-uniform angular occupancy. Similar as the study of Go Dynamics, we sampled uniform neural states on a ring in the response plane, then let them evolve through response epoch. Note that it's the averaged response neural activity will be decoded into color (Equation 7), therefore instead of showing neural states at the end of the response epoch, we show representative response neural states which are the temporal averaged neural states across the whole response epoch (Figure 7D). Results suggested that angular motion of neural states also happens when the environment prior σ_s is small (Figure 7E). This suggested that angular motion during the response epoch is a possible reason leading to larger angular occupancy in the delay plane.

The above two dynamic mechanisms should not be too surprising. The go and response epochs themselves require a duration of time, necessitating the maintenance of information throughout the decoding phase, thus providing an opportunity for neural dynamics to play a role. However, it is important to recognize that the dynamic system during the go and response epochs are distinct that of delay epoch. The go pulse acts as an input to the neural system, prompting a shift in neural states during the go epoch. Consequently, the inputs and outputs during the decoding phase differ from that in the delay phase, and hence, it should not be regarded as merely a trivial extension of the delay dynamics.

Not only the dynamical evolution of recurrent state during the decoding phase matters, how the recurrent state maps to an output color (including the mapping from recurrent neuron to response neuron and population method for translating response neural activities to an output color, Equation 3 and 7) is also important. This is called Readout and does not have a time factor hence it is not a dynamics process. As an extreme example, a trivial Readout can map all possible recurrent neural states (in the response plane) into a single color, hence the angular occupancy would be highly biased.

To examine the Readout's characteristics, we sampled a grid mesh of points within the response plane. Then, utilizing the readout matrix along with the bias current (as elaborated in Equation 3 and the population vector method in Equation 7), we transformed these mesh points into corresponding colors (illustrated in Figure 7G). The resulting data (Figure 7H) reveals that the Readout itself exhibits a bias, mapping larger angular spaces to common colors. This bias becomes increasingly pronounced with a smaller environmental prior σ_s (Figure 7I). Interestingly, RNNs trained on shorter response epoch demonstrated a stronger readout bias (SI Figure 3). This suggests that in more realistic scenarios, where the agent has limited time to respond, the readout mechanism may play a more dominant role in shaping the decoding strategy to reduce memory error.

Overall, we suggest that angular motion during the go and response epoch, and the biased readout process (including bias current in Equation 3) are the possible reasons for larger angular occupancy for common color in the delay plane.

Discussion

Our results highlight the importance of decoding strategy in memory error reduction. Smaller memory error for a color can be achieved by allocating larger state space occupancy (or angular occupancy in this paper). Further analysis revealed that biased state space occupancy is due to both dynamics during the decoding process (go and response epochs) and a biased readout from recurrent neural activity to output color.

The logic of this paper resembles a previous human/animal behavioural experiment[7]: training/testing the agent to perform a memory task within a specific prior color distribution, observing a reduction in memory errors for some input colors, and subsequently proposing and/or testing potential neural mechanisms for this error reduction. Conversely the suggested memory error reduction mechanisms can be more universal than merely adapting to environmental priors. For instance, in Figure 6, we demonstrated that RNNs trained on the same prior but with different noise levels exhibit similar mechanisms for reducing memory errors. In a broader context, mechanisms for reducing memory errors might arise when there is a high demand for accurate retention of information. One potential example is reinforcement learning[26]. The importance of information values is weighted by their associated rewards. Consequently, we further hypothesize that RNNs will adopt similar mechanisms for reducing memory errors (attractor dynamics and a larger state space occupancy) for high reward values. This hypothesis in future could be tested in physiology experiments [7,27–29] by appropriately assigning rewards to different information values while training animals to perform delayed-response tasks, or could be tested on RNN models using reinforcement learning[30,31].

Concretely, this paper proposes several experimental testable phenomena. First, when the input value is important (i.e., high-prior color or high-reward value), the neural population dispersion (measured by the variance of repeated trials) should be smaller. Second, state space occupancy for important values should be larger. The state space occupancy can be measured similar to the numerical procedure described here. Specifically, test the animal on multiple trials with different input colors and collect the neural population states at the end of the delay. These neural states, along with the animal's output color, provide a mapping from neural state space in the delay to the output color. This mapping is determined by the decoding phase hence can be used for studying the decoding strategy. With the mapping from neural state space to output color, numerical differentiation can then be computed. State space occupancy for important colors (high-prior or high-reward) can be compared with other colors. Finally, the approximate formula (Figure 5A-E) can also be tested. For example, train the animal to perform delayed-response tasks in different environmental priors[7,32], and then measure the dynamic dispersion, angular occupancy, and mean bias correction separately. Theoretical predictions can be compared with the animal's experimental memory error.

In certain physiology experiments, the neural population state resides within a low-dimensional (2D or 3D) manifold and employs a ring-like structure to represent information [11,27,33], akin to the ring structure discovered in our trained RNNs. In such cases, the state space occupancy can be computed by determining the angle of the neural population state. However, we are aware that in many cases, neural manifolds exist in high dimensions, such as a highly curved trajectory within a high-dimensional space[2,4]. In these instances, state space occupancy does not rely on angular occupancy but rather on arc-length occupancy. Arc-length occupancy means the arc length used to represent a unit change of information. Measuring the arc-length occupancy of a highly curved trajectory can be challenging. Yet this challenging can be eased by the concepts and methods advanced in the recent geometric framework[34–38]. Various nonlinear dimensional reduction

methods[38] (e.g., CEBRA[39], UMAP[40]) can project the manifold into a few latent dimensions for better visualization. Additionally, methods like SPUD[41] or simple kernel smoothing[32] enable the direct parameterization of a high-dimensional curve by its arc length, facilitating the measurement of arc length occupancy for each information value. In future, it would be interesting to explore the memory error reduction mechanisms in high-dimensional and more complex task scenarios.

Beyond memory error reduction mechanisms, how RNNs adapt to the entire environmental prior is an interesting problem. Adaptation is not simply about reducing the memory error for all colors. As observed in Figure 2B, also in previous works[7,8,42], some low-prior colors exhibit larger memory errors. Thus, there is a trade-off for the RNN: how much error should be reduced for high-prior colors and how much should be increased for low-prior colors. This trade-off is unavoidable due to the neural mechanism. Introducing attractors to reduce memory error for common colors unavoidably leads to biased (or truncated) errors for nearby colors[7,8]. In the angular occupancy (Figure 4) part, enlarging common color's angular occupancy will lead to smaller occupancy for other colors. How do RNNs balance memory errors among different colors remains unclear. Possible explanations can come from the Bayesian inference framework[18,43]. For example, Sohn et al.[32] assumes that the agent adapts to the environmental prior by making Bayesian inference in each trial. Environmental prior is the prior function, noise distribution is the likelihood function. Based on the prior and likelihood, they proposed that the agent needs to output a memorized information value (in that paper was time duration) which equals to the mean of the posterior distribution. This theory can make concrete prediction on the neural dynamics. Alternative theory suggests the network can be a generative model for reproducing the environmental prior[14]. RNN trained in this paper provides a ready-to-use models for theory pre-testing.

Materials and Methods

RNN architecture

The RNN consists of 12 perception input neurons, 1 go input neuron, 12 response output neurons and 256 fully connected recurrent neurons (Fig. 1A). The dynamic equation of the RNN is[44]

$$\boldsymbol{x}_t = (1-\alpha)\boldsymbol{x}_{t-1} + \alpha\left(\boldsymbol{W}^{rec}\boldsymbol{r}_{t-1}^{rec} + \boldsymbol{W}^{in}\boldsymbol{u}_t + \sqrt{2\alpha^{-1}\sigma_{rec}^2}\boldsymbol{\epsilon} + \boldsymbol{b}\right), \tag{2}$$

where $\boldsymbol{x}_t$ is a 256-dimensional vector representing the activities/state of recurrent neurons (neural state) at time step t, and $\alpha = \Delta t / \tau$, where Δt is the time length for each computational step, τ is the neuron membrane time constant. In this study, we set $\Delta t = 20$ ms and $\alpha = 1$, which is the same choice as in previous work[22,44,45]. The change of neural state x_{t+1} subjects to 4 factors: (1) The inputs from the other recurrent neurons, $\boldsymbol{W}^{rec}\boldsymbol{r}_{t-1}^{rec}$, where $\boldsymbol{r}_t^{rec} = \tanh(\boldsymbol{x}_t)$ and $\boldsymbol{W}^{rec}$ is the recurrent weights but the diagonal elements fixed to 0 (self-connections are not allowed)[44] (2) The inputs from perception and go neurons, $\boldsymbol{W}^{in}\boldsymbol{u}_t$, where $\boldsymbol{u}_t$ consists of 12 components for the perception neurons and 1 component for the go neuron, and $\boldsymbol{W}^{in}$ is the input weights. During the perception epoch, the perception neurons is entangled with an addictive gaussian noise with standard deviation $\sigma_x = 0.2$. (3) The intrinsic recurrent noise in the RNN, $\sqrt{2\alpha^{-1}\sigma_{rec}^2}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a 256-dimensional noisy vector drawn from the standard normal distribution at each time step, and $\sigma_{rec} = 0.2$ is the noise strength used in all main figures, except in Figure 6 where we inspected different noise strengths. (4) A constant bias current vector $\boldsymbol{b}$.

The activities of the 12 response neurons are read from the $\boldsymbol{r}_t^{rec}$,

$$\boldsymbol{z_t} = \boldsymbol{W}^{out}\boldsymbol{r}_t^{rec} + \boldsymbol{b}^{out}. \tag{3}$$

RNN training

At the start of each trial, the recurrent neural state is initialized with small Gaussian noise. During the fixation epoch, the RNN does not receive any inputs (i.e., $\boldsymbol{u}_t = 0$). In the perception epoch, a color ϕ_s is drawn from a prior distribution,

$$P_s(\phi_s) = \frac{1}{4}\sum_i \mathcal{VM}(\phi_s - \mu_i; \sigma_s^2), \tag{4}$$

which is the sum of four von Mises functions (circular version of the normal distribution) with mean values at $\mu_i \in \{40°, 130°, 220°, 310°\}$,

$$\mathcal{VM}(\phi_s - \mu_i; \sigma_s^2) = \frac{1}{2\pi I_0(1/\sigma_s^2)}\exp\left[\cos(\phi_s - \mu_i)/\sigma_s^2\right], \tag{5}$$

where σ_s is a free parameter to control the "width" of the prior distribution, and $I_0(\cdot)$ is the modified Bessel function of order 0 for normalization purpose. Units of ϕ_s, μ and σ_s are converted from degree to rad before feeding into the von Mises function. The color stimulus drawn from the above prior distribution is sensed by the 12 perception neurons, whose tuning curves are also von Mises functions $\mathcal{VM}(\phi_s - \mu_{p,i}; {\sigma_p}^2)$ with $\mu_{p,i} = \frac{360°}{12}i$ and $\sigma_p = \frac{1}{2}\frac{360°}{12}$ for $i = 0, 1, \ldots, 11$ (Fig. 1A). Therefore, in the perception epoch, the perception components of the input vector $\boldsymbol{u}_t$ are $u_{t,i} =$

$\mathcal{VM}\left(\theta_s - \mu_{p,i}; {\sigma_p}^2\right) + \epsilon_x$ where ϵ_x is a gaussian noise with standard deviation $\sigma_x = 0.2$, and the go component is 0 ($u_{t,12} = 0$).

Next, during the delay epoch, all inputs are 0. The delay length is drawn from an uniform distribution U(0 ms, 1000 ms) in each trial. Then, in the go epoch, we set $u_{t,12} = 1$ to indicate a go cue and the rest of inputs and the outputs are 0. Finally, in the response epoch, the 12 response neurons need to reproduce the neural activity of the perception neurons in the perception epoch, i.e. $\tilde{\boldsymbol{z}}_{t,i} = \mathcal{VM}\left(\phi_s - \mu_{p,i}; {\sigma_p}^2\right)$ for $i = 0, 1, \ldots, 11$.

The loss function is

$$Loss = \frac{1}{T_{tot}} \sum_t m_t \left[|\boldsymbol{z_t} - \tilde{\boldsymbol{z}}_t|^2 + \frac{1}{N_{rec}} (\beta T |\boldsymbol{W}_{rec}|^2 + \gamma |\boldsymbol{r}_{rec}(x) \ + \ 1|^2) \right], \tag{6}$$

where m_t is the mask that is 0 in the fixation epoch and 1 in the rest epochs, T_{tot} is the trial time length, N_{rec} is the number of recurrent neurons, β and γ in the regularization terms constrain the recurrent weights and neural firing rates. $\tilde{\boldsymbol{z}}_t$ is the target response output. It is zero across perception, delay and go epochs, and is the target response $\tilde{\boldsymbol{z}}_{t,i} = \mathcal{VM}\left(\phi_s - \mu_{p,i}; {\sigma_p}^2\right)$ during the response epoch. This loss function is minimized by gradient descent using the Adams optimizer[46].

Directly training the RNN to optimize the above loss function is difficult, we employed the progressive training protocol[24]: First, we trained the RNN with no noise, no regularization terms, uniform prior distribution, and delay epoch length fixed to 0. Secondly, we retrained the RNN with delay epoch length drawn from U(0 ms, 1000 ms). Thirdly, the RNN is retrained considering the noise and regularizations. The resulting model is called the pretrained RNN. Finally, we retrained the pretrained RNN with the desired prior distribution. If the desired prior distribution is still a uniform prior, then the trained RNN was called as uniform RNN (approximately equivalent to the prior $\sigma_s = 90°$), otherwise was called biased RNN.

Mapping response neural activity into color

The actual response neural activity can be mapped back to color by the population method. The mapped color is also called the RNN's output color. More precisely, the response neural activity z_t was firstly taking averaged within middle time window of the response epoch (60 ms to 140 ms, note the total response epoch is 200 ms). Next, the averaged response neuron activity $\bar{z}$ was converted into the output color by a populational vector method[44,47]

$$\hat{\phi} = ang\left(\sum_{m=1}^{N=12} \bar{z}_m \, e^{i\mu_{p,m}} \right), \tag{7}$$

where $ang(\cdot)$ means taking the angle of a complex number, i here is a unit imaginary number, and $\bar{z}_m$ is the mth components of vector $\bar{z}$, i.e. the mth perception neuron. Finally, $\mu_{p,m} = \frac{360°}{12} m$, which is set to be align with the perception tuning mean.

Removing outlier RNNs

Group analysis means computing and analyzing each RNN's metrics (e.g., memory error, entropy etc.) within a group. Due to the stochastic nature of the training process, some RNNs exhibited significantly different metric values compared to others, this may be due to a failure of training. To mitigate the influence of these outliers, this paper employs a criterion for exclusion: any RNN whose measured metric exceeds 1.5 times the interquartile range (IQR) above the third quartile or below the first quartile is removed from the analysis. This exclusion is specifically applied in the results presented in Figures 2, 4E, and 7, as indicated by the phrase 'outlier RNNs removed'.

Identifying fix points

The dynamic equation of the RNN, Eq. (1), can be reformulated in the form of

$$\boldsymbol{x}_t - \boldsymbol{x}_{t-1} = F(\boldsymbol{x}_{t-1}), \tag{8}$$

which defines the velocity field $\boldsymbol{v}(\boldsymbol{x}) = F(\boldsymbol{x}) \,/\, \Delta t$. In this study, we set $\Delta t = 20$ ms. The global minimum points of speed $v(\boldsymbol{x}) = |\boldsymbol{v}(\boldsymbol{x})| = 0$ are the fixed points. Specifically, we ran the trained RNN with 500 trials with stimuli equally distributed from $0°$ to $360°$. At the beginning of the delay epoch in these trials, neural states locate near the heteroclinic cycle (ring structure in the delay epoch), hence they are set to be the initial searching points. Then, starting from these initial neural states, we searched for the global minima of speed using the gradient descent, where the independent variable is the position $\boldsymbol{x}$. Due to the imperfection of the gradient descent algorithm, we may end up with local minima. Nevertheless, these local minimum points are similar to the fixed points[48], hence they are also called fixed points in this paper. For each fixed point, we computed the eigenvalues and eigenvectors of its Jacobian matrix $J = \frac{\partial F(x)}{\partial \boldsymbol{x}}$. A fixed point is called an attractor if its largest eigenvalue is negative, while it is called saddle point if some eigenvalues are positive. The eigenvectors of the positive eigenvalues are projected and shown on the $PC_1 - PC_2$ plane (Fig. 4). Only fixed points near the heteroclinic cycle were shown.

Decoding a neural state in the delay epoch

We built an RNN decoder to decode a neural state $\boldsymbol{x}_t^{rec}$ to its corresponding color. This RNN decoder faithfully represents the decoding process of the RNN. Specifically, an RNN decoder is the same as the original trained RNN, but its recurrent neural state is initialized as the to-be-decoded neural state. Then, the RNN decoder runs directly through the go and response epoch. Response neural activities in the response epoch were averaged. Averaged response was then decoded to an output color by the population vector method (Equation 7). This RNN decoder was used, for example, to decode attractors color (Figure 4D).

We also used the RNN decoder to decode the PC1-PC2 plane of the delay. The RNN was firstly ran through 1000 trials with uniformly distributed input color. Neural states during the delay were collected, used for fitting the PC1-PC2 plane, also called the delay plane. Next, a grid of mesh points (50 times 50) was sampled in the delay plane, centered at the delay plane center. Each of these mesh points were then PCA inversed back the original high-dimensional space, finally decoded by the RNN decoder (Figure 4G).

We also used the RNN decoder to find the relation between the angle of the delay plane and the color. Similar as above, a RNN was firstly ran through 1000 trials with uniformly distributed input color. Neural states at the end of delay (delay length is fixed as 800 ms) were collected, for fitting the PC1-PC2 plane. In addition, the average radius of all collected 1000 neural states were also computed, by averaging neural states' distances to the PC1-PC2 plane center. Using this radius,

dense points (1000 points) of a ring were resampled, each of these resampled points have different angle. Finally, each of the resample points were decoded by the RNN decoder. This provides a mapping from the angle of the PC1-PC2 plane to the color (Figure 4H).

Cross-decoding experiment

Cross-decoding involves using one RNN model (model 1) to establish the delay state, which is then decoded by another RNN model (model 2). Initially, model 1 is selected from one of two categories: Biased RNN or Uniform RNN. This model undergoes 500 trials with a fixed input of the common color, and the neural states at the end of these delays are collected. These end-of-delay neural states are then used to set the recurrent neural states of model 2, sampled from the second category (can be the same category as model 1). Model 2 processes these states through its decoding phase, outputs colors, and allows for the computation of memory error. This process is repeated to obtain memory error data for 50 pairs from each category combination, including Bias & Bias, Bias & Uniform, Uniform & Bias, and Uniform & Uniform, with the first category representing model 1 and the second model 2.

A key challenge in this process is the 'neuron-matching problem' which involves determining how to match neurons in model 2 with those in model 1. Our approach to neuron matching is based on comparing the neural preferred color rank. The preferred color for a neuron in an RNN model is determined as the following. Dense neural states along a ring on the PC1-PC2 delay plane were sampled; then decoded into output colors. This provides a delay-neural-states-to-color mapping, which also serves as a neural-activation-to-color mapping. The preferred color for each neuron is defined as the color that maximizes its neural activation.

We calculated the preferred colors of all recurrent neurons in model 1 and ranked them from 0 degrees (smallest) to 360 degrees (largest), repeating this procedure for model 2. Neuron pairing is achieved by matching these ranks. Thus, the prepared delay neuron state (comprising 256 neurons) in model 1 can be exactly matched to the recurrent neural state (also 256 neurons) in model 2. Subsequently, model 2 undergoes the decoding phase to compute the memory error.

A Taylor expansion approximation to the memory error

The squared memory error of a fixed common color ϕ_c is

$$\epsilon^2(\phi_c) = \sum_{i=1}^{N} (\phi_i - \phi_c)^2 / N, \tag{9}$$

where N is the number of trials, ϕ_i is the RNN's reported color in trial i. Approximately the RNN encodes color by the angle of the delay plane θ, so output color is a function of the neural state's angle $\phi(\theta)$. Denoting the angle encoding common color as θ_c, assumes during delay the actual neural state angle doesn't deviate from θ_c too much, then we can Taylor expand $\phi(\theta) \approx \phi_c + \frac{d\phi}{d\theta}|_{\phi_c}(\theta - \theta_c)$. Substituting this back to equation (9) yields

$$
\begin{aligned}
\epsilon^2(\phi_c) &= \sum_{i=1}^{N} \left(\phi_i - \phi_c\right)^2 / N \\
&\approx \left(\frac{d\phi}{d\theta}|_{\phi_c}\right)^2 \sum_{i=1}^{N} (\theta_i - \theta_c)^2 / N \\
&= \left(\frac{d\phi}{d\theta}|_{\phi_c}\right)^2 \sum_{i=1}^{N} [(\theta_i - \bar{\theta}) + (\bar{\theta} - \theta_c)]^2 / N \\
&= \left(\frac{d\phi}{d\theta}|_{\phi_c}\right)^2 \sum_{i=1}^{N} \frac{(\theta_i - \bar{\theta})^2}{N} + \left(\frac{d\phi}{d\theta}|_{\phi_c}\right)^2 (\bar{\theta} - \theta_c)^2,
\end{aligned}
\tag{10}
$$

were $d\phi/d\theta|_{\phi_c}$ is the reciprocal of the angular occupancy at common color, $\sum_{i=1}^{N}(\theta_i - \bar{\theta})^2 / N$ is the dynamic dispersion, and the last term describes a correction from the actually neural state mean angle to the common color angle.

Numerical procedure for testing the Taylor expansion approximation.

To test this theory, we computed the dynamic dispersion, angular occupancy and mean bias correction. To compute the dynamic dispersion, we ran each of the RNN 5000 trials with fixed input color as common color (delay length 800 ms). Neural states at the end of the delay were collected, and their corresponding angles θ_i were computed. Outlier angles (1.5 IQR below the first quantile or above the third quantile) were removed. Dynamic dispersion was measured as the variance of θ_i. To compute angular occupancy, similar as Figure 4H (see Methods), dense points along a ring were sampled in the delay plane. Each of these points was decoded by the RNN decoder, hence we obtained the angle-color mapping. Numerical differentiation was then performed to compute the angular occupancy. Finally, mean bias correction was computed as written in the Equation (10).

The resulting memory error computed from all these terms (Equation 10) is called theoretical prediction. To emphasize the importance of decoding strategy (i.e. angular occupancy), as a comparison, we also computed theoretical prediction but setting angular occupancy to 1. For the experimental RNN's memory error, for each condition (e.g. environmental prior or noise level), 50 RNNs were used. Each RNN was ran on 5000 trials. After removing the outliers' trial errors, each RNN's experimental memory error is the trial error standard deviation (squared root of Equation 9).

Decoding neural states in the response epoch

A neural state during the delay can decoded into an output color by using the RNN decoder which is continuing running the RNN through go and response epochs. Similarly, a neural state in the response epoch can also be decoded into an output color by continuing running the RNN. Since the response is already the last epoch, hence the "continuously running" only means reading the (recurrent) neural state out to the response neuron activity, by using RNN's readout weight $\boldsymbol{W}^{out}$ and bias current $\boldsymbol{b}^{out}$ (Equation 3). Decoding one single neural state in the response plane will only yield one singe response neurons' population response. This single population response was then

translated into an output color using populational vector method (equation 7). Note that no dynamics is involved since no time component here. This decoding procedure is also called readout decoder. Readout decoder's properties is fully determined by the readout weight and bias currents.

To inspect the properties of readout weight, we used the readout decoder to decode mesh points on the response PC1-PC2 plane. Specifically, each RNN was run through 1000 trials, neural states during the response epochs were collected for fitting the response plane (PC1-PC2 plane). After fitting, we mesh grid points on the response plane. They were further decoded by the readout weight.

Similarly, we can also inspect the relation between angle and the color in the response plane. The procedure is analogous to what was described in the "decoding a neural state in the delay epoch" session, except that the delay plane was replaced by the response plane, and readout decoder was used.

References

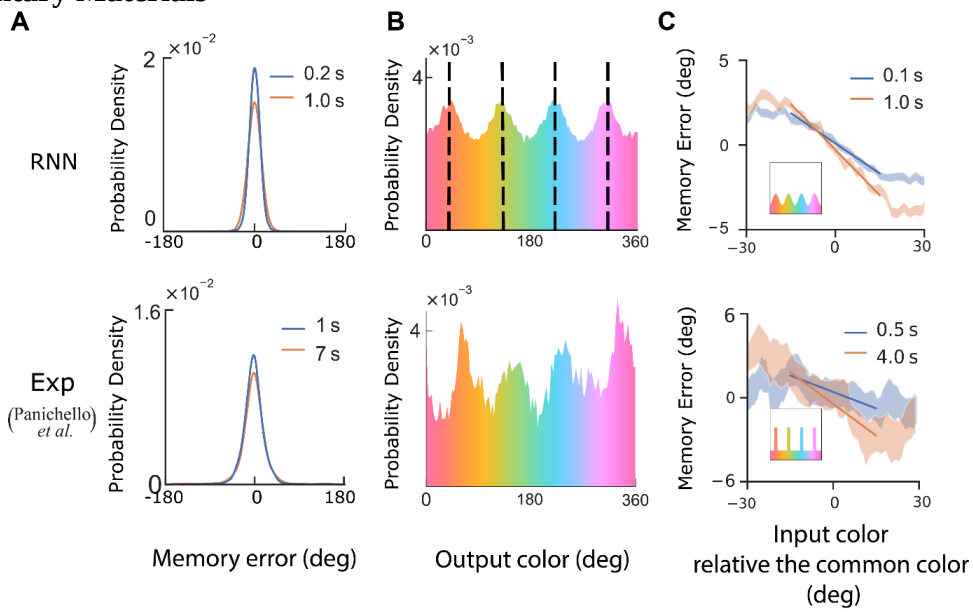
1. Burak, Y. & Fieted, I. R. Fundamental limits on persistent activity in networks of noisy neurons. *Proc. Natl. Acad. Sci. U. S. A.* **109**, 17645–17650 (2012).
2. Khona, M. & Fiete, I. R. Attractor and integrator networks in the brain. *Nat. Rev. Neurosci.* **23**, 744–766 (2022).
3. Renart, A., Song, P. & Wang, X. J. Robust spatial working memory through homeostatic synaptic scaling in heterogeneous cortical networks. *Neuron* **38**, 473–485 (2003).
4. Shaham, N. & Burak, Y. Slow diffusive dynamics in a chaotic balanced neural network. *PLoS Comput. Biol.* **13**, 1–26 (2017).
5. Seeholzer, A., Deger, M. & Gerstner, W. *Stability of working memory in continuous attractor networks under the control of shortterm plasticity*. *PLoS Computational Biology* vol. 15 (2019).
6. Eissa, T. L. & Kilpatrick, Z. P. Learning efficient representations of environmental priors in working memory. *bioRxiv* 2022.07.05.498889 (2022).
7. Panichello, M. F., DePasquale, B., Pillow, J. W. & Buschman, T. J. Error-correcting dynamics in visual working memory. *Nat. Commun.* **10**, 1–11 (2019).
8. Kilpatrick, Z. P., Ermentrout, B. & Doiron, B. Optimizing working memory with heterogeneity of recurrent cortical excitation. *J. Neurosci.* **33**, 18999–19011 (2013).
9. Xiong, H.-D. & Wei, X.-X. Optimal encoding of prior information in noisy working memory systems. *Conf. Cogn. Comput. Neurosci. (CCN, 2022)* (2022) doi:10.32470/ccn.2022.1162-0.
10. Chaudhuri, R. & Fiete, I. Computational principles of memory. *Nat. Neurosci.* **19**, 394–403 (2016).
11. Murray, J. D., Bernacchia, A., Roy, N. A., Constantinidis, C., Romo, R. & Wang, X. J. Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proc. Natl. Acad. Sci. U. S. A.* **114**, 394–399 (2017).
12. Inagaki, H. K., Fontolan, L., Romani, S. & Svoboda, K. Discrete attractor dynamics underlies persistent activity in the frontal cortex. *Nature* **566**, 212+ (2019).
13. Darshan, R. & Rivkind, A. Learning to represent continuous variables in heterogeneous neural networks. *Cell Rep.* **39**, 110612 (2022).
14. Kilpatrick, Z. P. Synaptic mechanisms of interference in working memory. *Sci. Rep.* **8**, 1–20 (2018).
15. Saxena, S. & Cunningham, J. P. Towards the neural population doctrine. *Curr. Opin. Neurobiol.* **55**, 103–111 (2019).
16. Haefner, R. M., Gerwinn, S., MacKe, J. H. & Bethge, M. Inferring decoding strategies from choice probabilities in the presence of correlated variability. *Nat. Neurosci.* **16**, 235–242 (2013).
17. McGinty, V. B. & Lupkin, S. M. Behavioral read-out from population value signals in primate orbitofrontal cortex. *Nat. Neurosci.* **26**, (2023).
18. Fritsche, M., Spaak, E. & de Lange, F. P. A bayesian and efficient observer model explains concurrent attractive and repulsive history biases in visual perception. *Elife* **9**, 1–32 (2020).
19. Ni, A. M., Huang, C., Doiron, B. & Cohen, M. R. A general decoding strategy explains the relationship between behavior and correlated variability. *Elife* **11**, 1–21 (2022).
20. Sadtler, P. T., Quick, K. M., Golub, M. D., Chase, S. M., Ryu, S. I., Tyler-Kabara, E. C., Yu, B. M. & Batista, A. P. Neural constraints on learning. *Nature* **512**, 423–426 (2014).
21. Williams, A. H., Kim, T. H., Wang, F., Vyas, S., Ryu, S. I., Shenoy, K. V., Schnitzer, M., Kolda, T. G. & Ganguli, S. Unsupervised Discovery of Demixed, Low-Dimensional Neural Dynamics across Multiple Timescales through Tensor Component Analysis. *Neuron* **98**, 1099-1115.e8 (2018).
22. Orhan, A. E. & Ma, W. J. A diverse range of factors affect the nature of neural

representations underlying short-term memory. *Nat. Neurosci.* **22**, 275–283 (2019).
23. Yang, G. R. & Wang, X. J. Artificial Neural Networks for Neuroscientists: A Primer. *Neuron* **107**, 1048–1070 (2020).
24. Chaisangmongkon, W., Swaminathan, S. K., Freedman, D. J. & Wang, X. J. Computing by Robust Transience: How the Fronto-Parietal Network Performs Sequential, Category-Based Decisions. *Neuron* **93**, 1504-1517.e4 (2017).
25. Bae, G., Olkkonen, M., Allred, S. R. & Flombaum, J. I. Why Some Colors Appear More Memorable Than Others : A Model Combining Categories and Particulars in Color Working Memory. *J. Exp. Psychol. Gen.* **144**, 744–763 (2015).
26. Botvinick, M., Wang, J. X., Dabney, W., Miller, K. J. & Kurth-Nelson, Z. Deep Reinforcement Learning and Its Neuroscientific Implications. *Neuron* vol. 107 603–616 (2020).
27. Panichello, M. F. & Buschman, T. J. Shared mechanisms underlie the control of working memory and attention. *Nature* **592**, 601–605 (2021).
28. Constantinidis, C., Franowicz, M. N. & Goldman-Rakic, P. S. Coding specificity in cortical microcircuits: A multiple-electrode analysis of primate prefrontal cortex. *J. Neurosci.* **21**, 3646–3655 (2001).
29. Funahashi, S., Bruce, C. J. & Goldman-Rakic, P. S. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *J. Neurophysiol.* **61**, 331–349 (1989).
30. Song, H. F., Yang, G. R. & Wang, X. J. Reward-based training of recurrent neural networks for cognitive and value-based tasks. *Elife* **6**, 1–24 (2017).
31. Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D. & Botvinick, M. Prefrontal cortex as a meta-reinforcement learning system. *Nat. Neurosci.* **21**, 860–868 (2018).
32. Sohn, H., Narain, D., Meirhaeghe, N. & Jazayeri, M. Bayesian Computation through Cortical Latent Dynamics. *Neuron* **103**, 934-947.e5 (2019).
33. Wolff, M. J., Jochim, J., Akyürek, E. G., Buschman, T. J. & Stokes, M. G. Drifting codes within a stable coding scheme for working memory. *PLoS Biol.* **18**, 1–19 (2020).
34. DiCarlo, J. J. & Cox, D. D. Untangling invariant object recognition. *Trends Cogn. Sci.* **11**, 333–341 (2007).
35. Chung, S. Y. & Abbott, L. F. Neural population geometry: An approach for understanding biological and artificial neural networks. *Curr. Opin. Neurobiol.* **70**, 137–144 (2021).
36. Jazayeri, M. & Ostojic, S. Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. *Curr. Opin. Neurobiol.* **70**, 113–120 (2021).
37. Barack, D. L. & Krakauer, J. W. Two views on the cognitive brain. *Nat. Rev. Neurosci.* **22**, 359–371 (2021).
38. Cunningham, J. P. & Yu, B. M. Dimensionality reduction for large-scale neural recordings. *Nat. Neurosci.* **17**, 1500–1509 (2014).
39. Schneider, S., Lee, J. H. & Mathis, M. W. Learnable latent embeddings for joint behavioural and neural analysis. *Nature* **617**, 360–368 (2023).
40. McInnes, L., Healy, J. & Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* (2018).
41. Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A. & Fiete, I. The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nat. Neurosci.* **22**, 1512–1520 (2019).
42. Eissa, T. L. & Kilpatrick, Z. P. Learning efficient representations of environmental priors in working memory. *PLoS Comput. Biol.* **19**, (2023).
43. Lange, R. D., Shivkumar, S., Chattoraj, A. & Haefner, R. M. Bayesian encoding and decoding as distinct perspectives on neural coding. *bioRxiv* **26**, 1–16 (2023).
44. Bi, Z. & Zhou, C. Understanding the computation of time using neural network models.

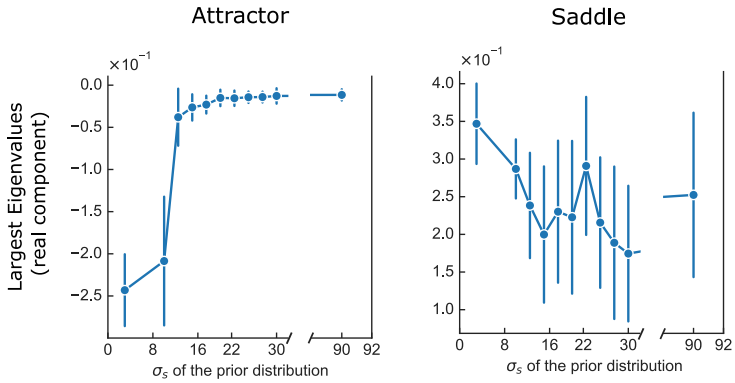
Proc. Natl. Acad. Sci. U. S. A. **117**, 10530–10540 (2020).

45. Wang, J., Narain, D., Hosseini, E. A. & Jazayeri, M. Flexible timing by temporal scaling of cortical responses. *Nat. Neurosci.* **21**, 102–112 (2018).
46. Kingma, D. P. & Ba, J. L. Adam: A method for stochastic optimization. *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.* 1–15 (2015).
47. Wimmer, K., Nykamp, D. Q., Constantinidis, C. & Compte, A. Bump attractor dynamics in prefrontal cortex explains behavioral precision in spatial working memory. *Nat. Neurosci.* **17**, 431–439 (2014).
48. Sussillo, D. & Barak, O. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput.* **25**, 626–649 (2013).

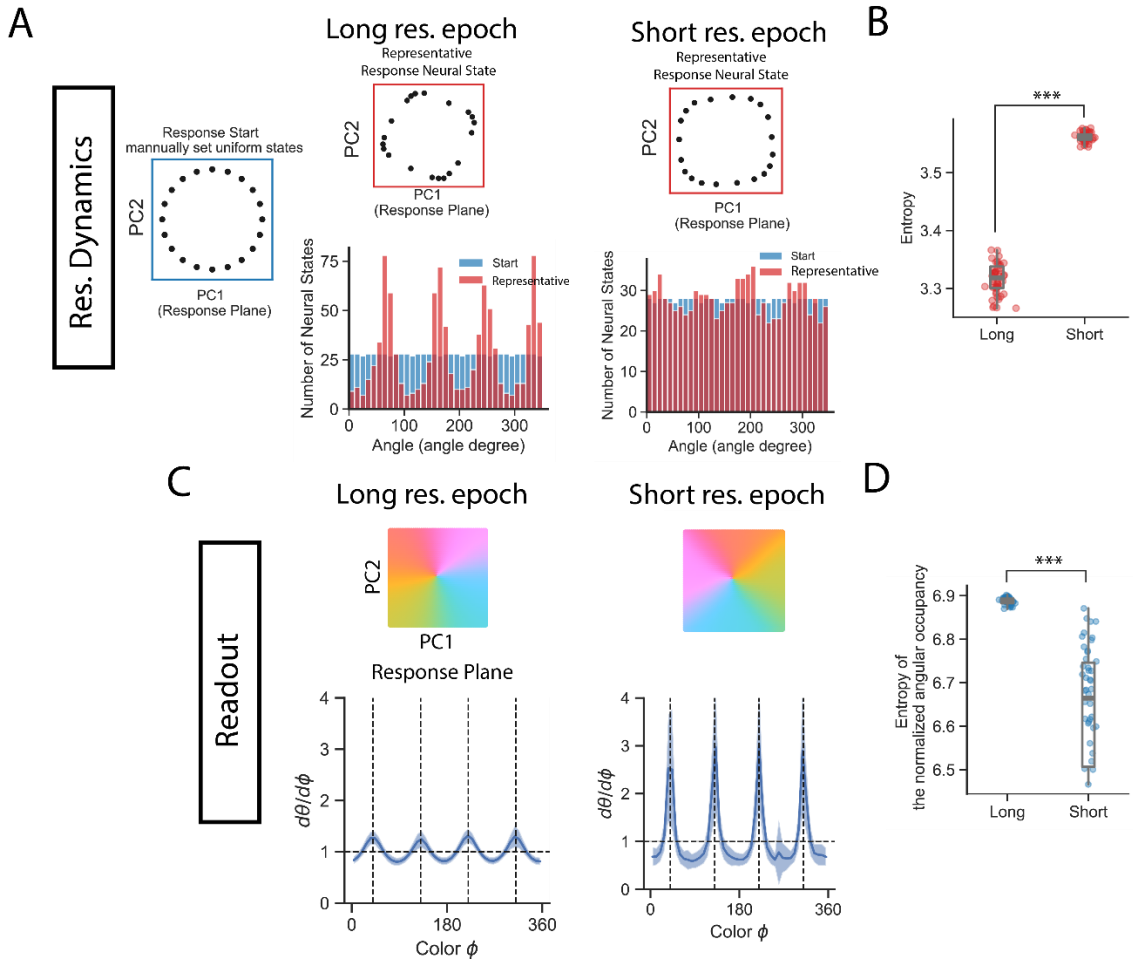
Supplementary Materials



SI Figure 1. Trained Biased RNN ($\sigma_s = 25°$) exhibited behavior features similar to the experiment. (**A**) The distribution of memory error. Each RNN was tested on 1000 trials with uniformly distributed input color. Memory error from 50 Biased RNNs were concatenated. Bottom: experimental data. (**B**) The distribution of output color when the input colors were sampled uniformly for each trial. 50 Biased RNNs results were concatenated, 1000 trials each. Black dashed lines indicate common colors in the training prior distribution. (**C**) Upper: memory error for each input color. 50 Biased RNN was used, each input color was run through 1000 trials for each of the Biased RNN. Bottom: human participants were trained on a biased prior distribution (inset). Their performance during the last third of the training trials is shown. The error band is the standard error of the mean across trials and RNNs. Experimental figures were plotted using the data published by Panichello et al[7].



SI Figure 2. Eigenvalues of the fixed points. (Left) We trained 50 RNNs for each prior distribution. For each RNN, we searched for the attractors (see Methods). The largest eigenvalue (real component) for each attractor is denoted as $v_{i,j}$, where i indicates the ith RNN and j indicates the jth attractor. The averaged largest eigenvalues across different attractors within the same RNN is denoted as $u_i = \frac{1}{N_j}\sum_j v_{i,j}$. Dots in the figure represent the mean of largest eigenvalues across different RNNs, $\frac{1}{N_i}\sum_i u_i$. Error bars indicate the standard deviation of u_i across different RNNs. (Right) Same as left but considering the largest real component of saddles. Both attractors and saddles become stronger (larger absolute values for the real parts) for narrower prior distributions.



SI Figure 3. RNN trained on a shorter response epoch has weaker response dynamics bias and stronger readout bias. Training prior $\sigma_s = 3°$. Computing details are identical to Figure 7 except that the RNN models were replaced by which trained with long/short response (res.) epoch. (**A**) Long/short response epoch: one example RNN trained with long (200 ms)/short (40 ms) epoch duration. Compared to RNN trained on long response epoch, RNN trained on short response do not have significant change of neural state during the response epoch. (**B**) Increasing entropy of the representative response neural state distribution also suggests that RNNs trained on short response epoch have less neural states' angular changes during the response. Each dot is one RNN's representative response neural state entropy (red histogram in panel A). ***: $p < 10^{-3}$ Wilcoxon signed-rank test. (**C**) RNN trained on short response epoch has more biased readout. Line shows the mean of 50 RNNs, and error band is the standard deviation. Dash lines: four common colors. (**D**) Entropy of angular occupancy, dot is an entropy of one RNN's angular occupancy (normalized). Outliers were not shown. ***: $p < 10^{-3}$ Wilcoxon signed-rank test.