PNAS
www.pnas.org

Supplementary Information for

Low dimensional dynamics for working memory and time encoding

Christopher J. Cueva, Alex Saez, Encarni Marcos, Aldo Genovesio, Mehrdad Jazayeri, Ranulfo Romo, C. Daniel Salzman, Michael N. Shadlen, Stefano Fusi

Corresponding authors: Christopher J. Cueva and Stefano Fusi
E-mail: ccueva@gmail.com and sf2237@columbia.edu

This PDF file includes:

Supplementary Materials and Methods

Neural data. Electrode recordings were from nonhuman primates as previously described (1, 5, 12, 13). For all datasets we only included correct trials in our analyses.

Vibrotactile discrimination task of Romo et al. (1): 160 PFC units were analyzed for the three second delay interval. Each unit was recorded for at least 10 trials for each value of f1 in the set [10 14 18 22 26 30 34] Hz. 139 PFC units were analyzed for the six second delay period. Each unit was recorded for at least 5 trials for each value of f1 in the set [10 14 18 22 26 30 34] Hz.

Context dependent trace conditioning task of Saez et al. (5): We analyzed units recorded for at least 50 trials in each of the four experimental conditions (context 1 or 2 and stimulus A or B) leading to 138 units in the Amygdala, 129 units in the OFC, and 102 units in the ACC.

Ready-set-go interval reproduction task of Jazayeri and Shadlen (12): 48 units were analyzed in LIP. Each unit was recorded for at least 20 trials with a minimum sample duration (interval between ready and set cues) of 1000 ms.

Duration discrimination task of Genovesio et al. (13): 148 units were analyzed in PFC. Each unit was recorded for at least 50 trials with a minimum S1 duration of 1000 ms.

Neural dimensionality. Our measure of neural dimensionality quantifies the *stable* component of the neural trajectory across trials (Figure S1). The stable component of the neural trajectory is important as it can be used for consistent computations by downstream neurons. Other common measures of dimensionality yield a large dimensionality even for random Gaussian noise data, with no consistent firing rate fluctuations across trials, as shown in Figure S2. For this data, the only consistent aspect is the mean and our 'trajectory reconstruction dimensionality' quantifies this data as zero dimensional.

To compute the cumulative neural dimensionality over time we used a cross-validation procedure to estimate the number of dimensions that gave us the greatest predictive power on firing rates from held out data. We first subdivided the time after stimulus offset into T nonoverlapping 100 ms intervals. For each timepoint, from 1 through $t = 1, \ldots, T$ that we wished to estimate the cumulative dimensionality we constructed a training and test matrix (FRtrain and FRtest) of size number-of-neurons $\times$ t containing firing rates after averaging spikes in 100 ms bins and across trials. FRtrain and FRtest contained averages from nonoverlapping sets of trials. If the neural activity was the same on every trial then FRtrain and FRtest would be equal and we would be able to predict the firing rates in FRtest perfectly from FRtrain. However, there is variability that is not shared between FRtrain and FRtest so FRtrain is not perfectly predictive. We also expect some variability in the neural trajectory of FRtrain is shared with FRtest so there is an optimal subspace of FRtrain that will yield the greatest prediction accuracy for FRtest. We estimated this subspace by first projecting FRtrain onto principal components 1 through k, sorted in order of descending variance so principal component 1 captures the most variance. We define the dimensionality as the number of principal components k that yields the greatest predictive accuracy for FRtest, i.e. that yields the smallest squared error between FRtest and the 'denoised' trajectory of FRtrain after projecting onto the first k principal components (Figure S1). We used 3/5 of the trials for training and the remainder for testing, repeating cross-validation 200 times. Single unit recordings were resampled to create 1000 trials for the FRtrain and FRtest matrices.

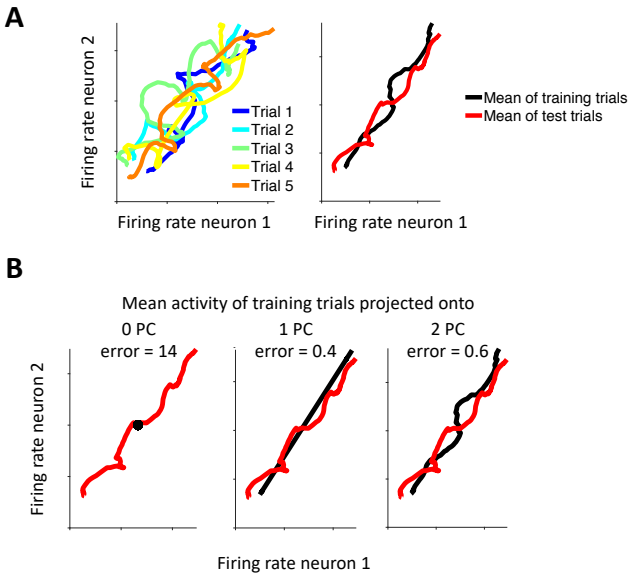

Fig. S1. Computing cumulative dimensionality. Our measure of neural dimensionality (trajectory reconstruction dimensionality) quantifies the stable component of the neural trajectory across trials. **(A)** The figure on the left shows five trials recorded from an example dataset consisting of two neurons. These trials are split into training and test groups, and then averaged within groups, as shown in the figure on the right. **(B)** Principal components are calculated from the mean training trajectory. The mean training trajectory is then projected onto the mean activity (left), first principal component (center), up to the maximum number, which in this example is the first two principal components (right). The mean squared error is then calculated between the training trajectory, after the projections, and the test trajectory for all timepoints. In this example, the projection of the training trials onto only one principal component yields the lowest error, i.e. is best able to predict the test trajectory, and so the dimensionality is one. This process is repeated for neural trajectories of varying lengths in order to assess the dimensionality as the firing activity evolves over time.

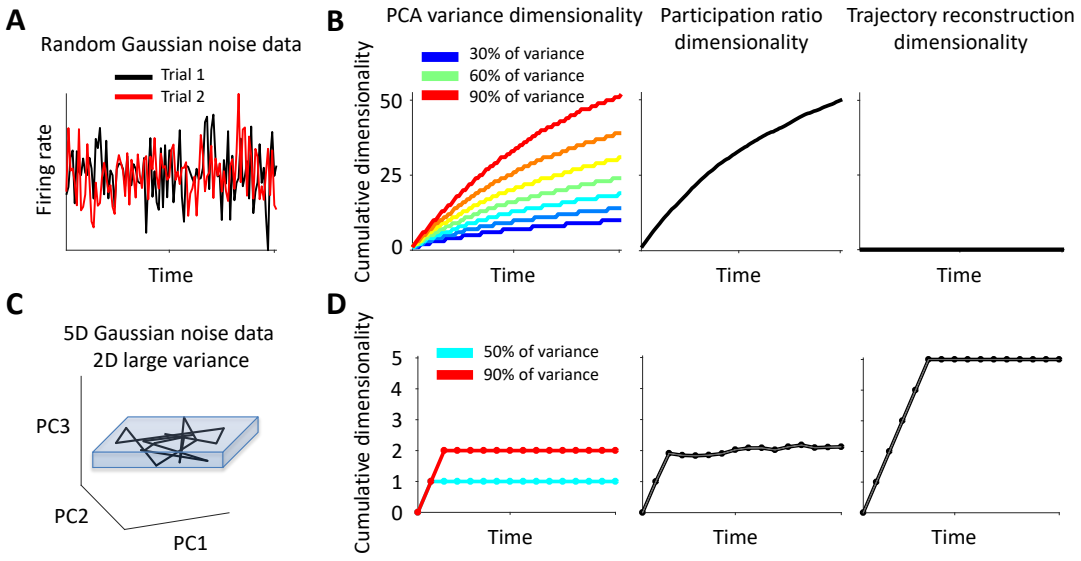

Fig. S2. Cumulative dimensionality computed with PCA variance, participation ratio, and trajectory reconstruction. The cumulative dimensionality is the dimensionality required to exaplain timepoints 1 through t where $t = 1, 2, 3, \ldots$ is increased from 1 to some maximum value. **(A)** Cumulative dimensionality of random data. The firing rate of each neuron, at each timepoint, on each trial is a new random sample from a Gaussian noise distribution. **(B)** In the figure on the left, the dimensionality is quantified as the number of principle components required to explain a fraction of the variance (e.g. 90% as shown in red). The center figure shows the cumulative dimensionality when using the Participation Ratio (2–4). Both methods show an increase in the dimensionality over time. However, by construction, there are no stable trajectories across trials that can be used for consistent computation. The only consistent aspect of the data is the mean and our 'trajectory reconstruction dimensionality' quantifies this data as zero dimensional. **(C)** The five dimensional synthetic data in this example has large variations along two dimensions and small variations along the other three. The neural trajectory of a single trial is shown after it has been projected onto a three dimensional space, using principal component analysis, capturing the largest variance in the trajectory over time. **(D)** The participation ratio incorrectly classifies the dimensionality as two (center), while the cumulative trajectory reconstruction dimensionality saturates at five (right). The trajectory reconstruction dimensionality tries to identify consistent components of the neural trajectory across trials, irregardless of the fraction of variance these components possess, and so correctly finds that five dimensions are needed to explain the trajectory of the full dataset.

Decoding time. To create the "two-interval time decode matrix", as detailed in Figure 1C, we first subdivide the time after stimulus offset into nonoverlapping 100 ms intervals. We take the vector of firing rates recorded from all neurons during a single interval (interval i) and train a logistic regression classifier to discriminate between this and another interval (interval j). We test the classifier on held-out trials and record the performance. This number, between 50% and 100%, from the binary classifier trained to discriminate intervals i and j is recorded in pixel (i,j) of the "two-interval time decode matrix." If the decode accuracy is 100% the pixel is colored yellow and if the decode accuracy is 50% the pixel is colored blue. We use 3/5 of the trials for training and the remainder for testing, resampling single unit recordings to create 10,000 trials for training and testing and then performing cross-validation 100 times to establish the final mean decode accuracy.

To quantify the timing uncertainty of the neural data at a given point in time we train a classifier to predict the time this recording was made (Figure S3A), and then use the spread of predictions when classifying firing rates from different trials as our measure of timing uncertainty. This multi-class classifier takes the firing rates from all neurons at a given timepoint (time is discretized in 100 ms bins) and predicts the time this recording was made. This is in contrast to the two-interval time decode analysis where the binary classifier only discriminates between two timepoints; the multi-class classifier attempts to predict the actual timepoint within the trial, e.g. 1000 ms after stimulus offset. We classify 10,000 trials (obtained through resampling single unit recordings) at each point in time yielding a distribution of predictions around the true value (Figure S3B). We calculate the standard deviation of this distribution (Figure S3C) and this is the metric for timing uncertainty. To classify trials we combine the pairwise binary classifications from the "two-interval time decode matrix" (Figure S3A), however, we obtain similar predictions with other multi-class classifiers. The chance level for the timing uncertainty is computed by training and testing the classifier on neural data with random time labels. To gain some intuition about the chance level distribution we can imagine the classifier predictions, X, are random and uniformly distributed over some interval T. We can compute various properties of X, for example, the expectation of X is $E[X] = 0.5 * (\min(T) + \max(T))$. Its standard deviation is $\sqrt{E[(X-t)^2]}$. If the chance-level-classifier guesses uniformly on the interval T then $E[(X-t)^2] = t^2 - t * (\min(T) + \max(T)) + ((\max(T)^3) - \min(T)^3)/(3 * (\max(T) - \min(T)))$, yielding a U-shaped curve for the timing uncertainty, $\sqrt{E[(X-t)^2]}$, as seen in the chance level curves of Figures 3-5. Intuitively, the chance level is U-shaped as a classifier with uniform, random predictions can make larger errors when the true value is at the edge of the interval. In practice, we don't use the analytic expression for $\sqrt{E[(X-t)^2]}$ but calculate this after training and testing a classifier on neural data that has the timepoint labels randomly shuffled. We use 3/5 of the trials for training and the

remainder for testing, resampling single unit recordings to create 10,000 trials for training and testing the classifier and then performing cross-validation 100 times.

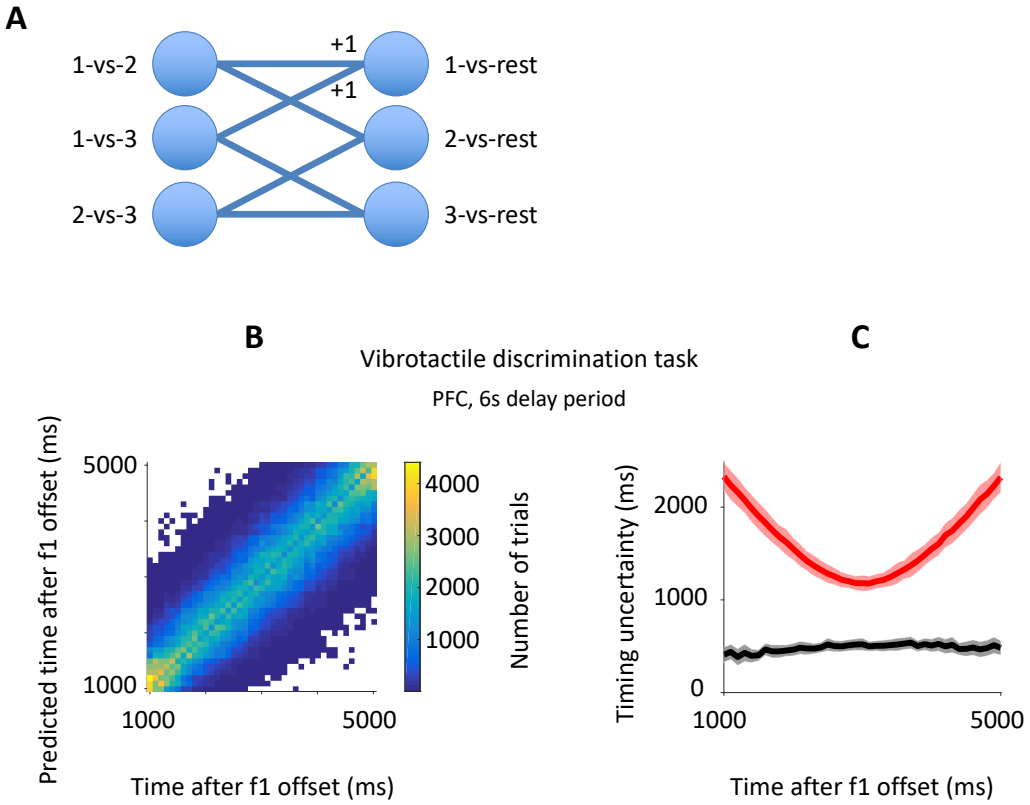

Fig. S3. Computing the timing uncertainty. **(A)** To classify neural data into one of T timebins we linearly combine results from the pairwise binary classifications obtained from the two-interval time decode. A schematic for $T = 3$ is shown here. For each datapoint, i-vs-j in [0,1] is the confidence of the binary classifier discriminating classes i and j in favor of the former class. The confidence of the classifier for the latter class is computed by j-vs-i = 1 - i-vs-j. To obtain the final classifier prediction we add the confidence votes. For example, the total confidence in favor of timepoint 1 (1-vs-rest) is the sum of 1-vs-2 and 1-vs-3. A similar calculation is performed for the total confidence in favor of the other timepoints (2-vs-rest and 3-vs-rest), and the timepoint with highest confidence is the output of the multi-class classifier. **(B)** To quantify the timing uncertainty at each point in time we calculate the predicted time (y-axis) relative to the actual elapsed time from stimulus offset (x-axis). The predicted time (cross-validated) is calculated for each of the 10,000 trials (obtained through resampling single unit recordings) and the results are shown as a heatmap. Data is from the delay interval of the vibrotactile discrimination task (1) **(C)** The standard deviation of the distribution in (B) is shown in black and is our metric for timing uncertainty. The chance level is shown in red. Error bars show two standard deviations.

Decode generalization. We first subdivide the delay period into T nonoverlapping 100 ms intervals, excluding the first few hundred milliseconds after stimulus offset to better study the intrinsic dynamics of the delay period without transient activity caused by the offset of the visual stimulus. To quantify the ability of a classifier to generalize to other timepoints we train a logistic regression classifier on a fraction of timepoints (from near 0 to 1) and then test its accuracy across the entire interval (see Figure S4). The plotted decode accuracy in Figure 8 is the mean of the classifier performance across the entire interval, when tested on trials that were not used during training. In the top row of Figure 8, the decoder classifies rewarded versus non-rewarded trials in the dataset of Saez et al. (5) during the interval from 500 ms to 1500 ms after stimulus offset. In the middle row, the decoder classifies high versus low frequency trials in the dataset of Romo et al. (1) during the interval from 500 ms to 3000 ms after stimulus offset (3s delay period) and 500 ms to 6000 ms (6s delay period). In the bottom row, the decoder classifies trials from the two patterns the network is trained to produce in Laje and Buonomano (6) during the interval from 200 ms to 1200 ms after stimulus offset. The chance level is computed by randomly shuffling these labels before training and testing the classifier. We use 3/5 of the trials for training and the remainder for testing, resampling single unit recordings to create 10,000 trials for training and testing the classifier and then performing cross-validation T times. Each time cross-validation is performed a new, unique set of timepoints are randomly chosen and used for training the classifier. Note that when only a single timepoint is used for training the classifier, we cycle once through each and every timepoint in the interval. Error bars show two standard deviations.

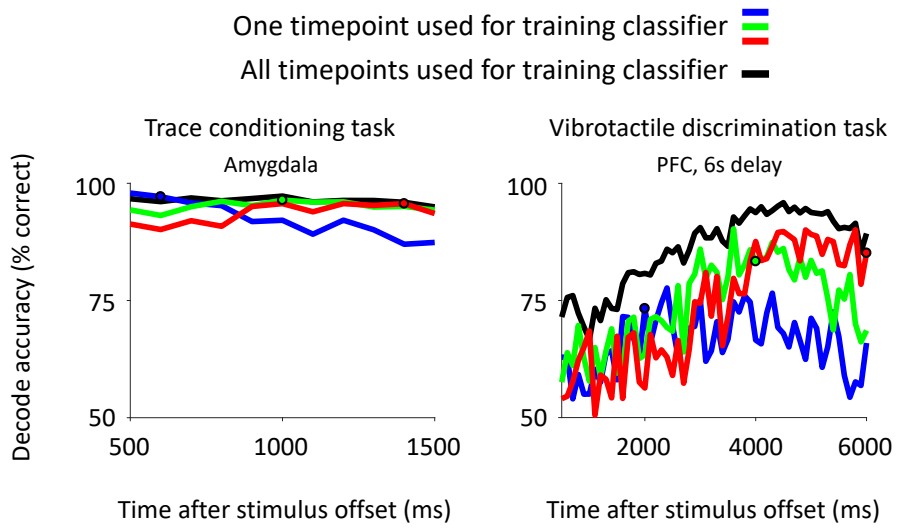

Fig. S4. Decode accuracy when training the classifier on one timepoint (blue, green, and red curves) and all timepoints (black curve). The decode accuracy is shown for a binary classifier trained to discriminate neural data from rewarded versus non-rewarded trials for the trace conditioning task (5) (Amygdala, left) and high versus low frequency trials for the vibrotactile discrimination task (1) (PFC, 6s delay period on right). Three examples are shown when the classifier is trained at a single timepoint (highlighted with circles); the classifier is trained at times 600 ms (blue), 1000 ms (green) and 1400 ms (red) after stimulus offset for the Amygdala data on the left and at times 2000 ms (blue), 4000 ms (green), and 6000 ms (red) after stimulus offset for the PFC data on the right. The decode accuracy is computed using trials that were not used for training the classifier. The leftmost datapoint in Figure 8, for example, summarizes the decode generalization when only a single timepoint is used for training the classifier, and is computed as the mean decode accuracy across the entire delay interval with the exception of the first 500 ms after stimulus offset (as shown here), for all possible choices of single training timepoints and iterations of cross-validation.

Model description. For each of the four working memory tasks we train a recurrent neural network (RNN) model. Our network models consist of a set of recurrently connected units ($N = 100$). The dynamics of each unit in the network $u_i(t)$ is governed by the standard continuous-time RNN equation (7, 8):

$$\tau \frac{dx_i(t)}{dt} = -x_i(t) + \sum_{j=1}^{N} W_{ij}^{\text{rec}} u_j(t) + \sum_{k=1}^{N^{\text{in}}} W_{ik}^{\text{in}} I_k(t) + b_i + \xi_i(t) \quad [1]$$

for $i = 1, \ldots, N$. The activity of each unit, $u_i(t)$, is related to the activation of that unit, $x_i(t)$, through a nonlinearity which in this study we take to be $u_i(t) = \tanh(x_i(t))$. Each unit receives input from other units through the recurrent weight matrix W^{rec} and also receives external input, $I(t)$, that enters the network through the weight matrix W^{in}. Each unit has two sources of bias, b_i which is learned and $\xi_i(t)$ which represents noise intrinsic to the network and is taken to be Gaussian with zero mean and constant variance. The network was simulated using the Euler method for T timesteps, with a step size of duration $\tau/10 = 10$ ms. To perform tasks with the RNN we linearly combine the firing rates of units in the network and use this as the output. The linear readout neurons, $y_j(t)$, are given by the following equation:

$$y_j(t) = \sum_{i=1}^{N} W_{ji}^{\text{out}} u_i(t) \quad [2]$$

The RNN for each working memory task has the same architecture but the network parameters W^{rec}, W^{in}, b and W^{out} are different for each task and adjusted to accomplish the task-specific transformation of time-varying inputs to time-varying outputs.

We optimized the network parameters W^{rec}, W^{in}, b and W^{out} to minimize the squared error in equation (3) between target outputs and the network outputs generated according to equation (2).

$$E = \frac{1}{MTN^{\text{out}}} \sum_{m,t,j=1}^{M,T,N^{\text{out}}} (y_j(t,m) - y_j^{\text{target}}(t,m))^2 \quad [3]$$

Parameters were updated with the Hessian-free algorithm (9) using minibatches of size $M = 500$ trials, i.e. 500 sequences of length T for each parameter update. In addition to minimizing the error function in equation (3) we regularized the input and output weights according to equation (4) to encourage the RNN to avoid decreasing the error, especially early during training, by making large but ultimately counterproductive changes to the input and output weights.

$$R_{L2} = \frac{1}{NN^{\text{in}}} \sum_{i,j=1}^{N,N^{\text{in}}} (W_{ij}^{\text{in}})^2 + \frac{1}{NN^{\text{out}}} \sum_{i,j=1}^{N^{\text{out}},N} (W_{ij}^{\text{out}})^2 \quad [4]$$

The parameters W^{out} and b were initialized to zero. W^{in} was initialized with random values drawn from a normal distribution with zero mean and variance $1/N^{\text{in}}$. W^{rec} was initialized as a random orthogonal matrix. Although the recurrent weight matrix at the end of training is not orthogonal this initialization helps with the problem of exploding and vanishing gradients (10).

For all four networks, the "firing rate" of each of the 100 units ($u(t)$) is stored every 100 ms and this activity is used for all subsequent analyses.

Inputs and outputs for the network: The RNN for the vibrotactile discrimination task of Romo et al. (1) is trained to report whether the frequency of the second stimulus (f2) is higher or lower than the frequency of the first (f1), and also to anticipate the time when f2 is presented. The RNN has one input that varies in magnitude to represent the frequency of f1 and f2, and two outputs; an output to indicate the choice for the binary discrimination, and an anticipatory-timer output that turns on before f2 onset (Figure S5). The inputs have a duration of 500 ms with amplitudes similar to Barak et al. (11) that linearly map the frequencies between 10-34 Hz to inputs between 0.2 and 1.8: input $= 0.2 + (1.8 - 0.2) * (f - 10)/(34 - 10)$ where f is the frequency in Hz. The RNN was trained on sequences of duration 15000 ms with successive presentations of f1 and f2 randomly chosen between 10 and 34 Hz. On each sequence the delay between f1 and f2 was fixed, selected uniformly between 200 and 7000 ms, while the intertrial interval between f2 offset and f1 onset was uniformly distributed between 200 and 1000 ms. The RNN output that performs frequency discrimination takes values of +1 if f2 $>$ f1 and -1 if f2 $<$ f1. This output is zero until f2 onset, whereupon it takes the appropriate nonzero value until f2 offset, at which time the amplitude returns to zero. The anticipatory-timer output is not constrained during the first f1/f2 pairing in a sequence, because the interval between f1 and f2 has not been established for this sequence. During subsequent inputs of f1/f2 the anticipatory-timer output takes a value of one 200 ms before f2 onset and then returns to zero immediately before f2 onset.

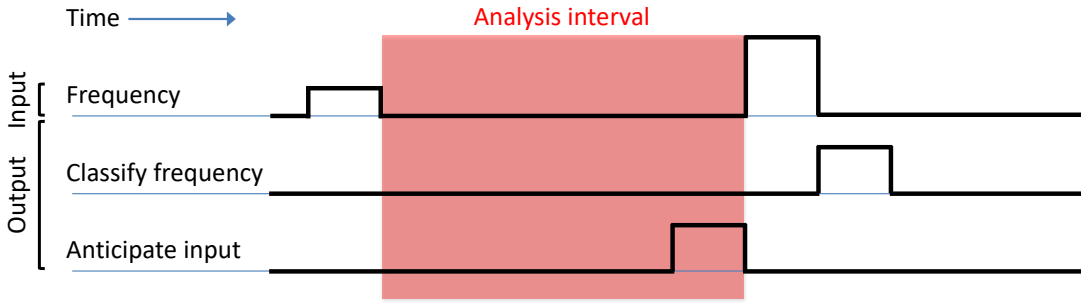

Fig. S5. RNN inputs and outputs for the vibrotactile discrimination task of Romo et al. (1). The RNN has one input representing vibrotactile frequency with the amplitude of an input pulse. After two successive frequency inputs the RNN reports whether the frequency of the second stimulus is higher or lower than that of the first by modulating an output to be +1 or -1 respectively. The RNN also anticipates the timing of the second frequency input by activating a second output before the onset of the stimulus. The interval we analyzed, after the offset of the first input and before the second, is highlighted in red.

The RNN for the context dependent working memory task of Saez et al. (5) has four inputs (stimulus A, stimulus B, reward, and the end-of-trial cue) and two outputs (reward predictive output and a no-reward predictive output) as shown in Figure S6. During context 1 stimulus A is followed by a reward and stimulus B is not rewarded. During context 2 the associations are reversed and stimulus B is rewarded while stimulus A is not rewarded. Context is not given to the RNN but must be inferred from the previous stimulus/reward pairing stored in the network's firing activity. The RNN indicates its knowledge of context by switching the appropriate output from zero to one, after stimulus offset, to indicate either an expected future reward or no reward. The inputs are presented serially, e.g. stimulus A, followed by the reward, followed by the end-of-trial cue, each having a value of one for 200 ms before returning to their baseline values of zero. The reward predictive output turns on immediately after the presentation of the stimuli the network thinks will be rewarded and stays on until the end-of-trial cue. The no-reward predictive output turns on immediately after the stimuli the network thinks will not be rewarded and stays on until the end-of-trial cue. The reward can either follow stimulus A (context 1) or stimulus B (context 2).

We trained the network using sequences of length 7000 ms with randomly switching contexts and intervals between events. An example trial is shown in Figure S7. During training, the interval between the end-of-trial cue and new stimulus was uniformly distributed between 0 and 1000 ms. The interval between the end of a stimulus and the reward, if present, was uniformly distributed between 0 and 1500 ms. The interval between the end of the reward and end-of-trial cue was uniformly distributed between 0 and 500 ms.

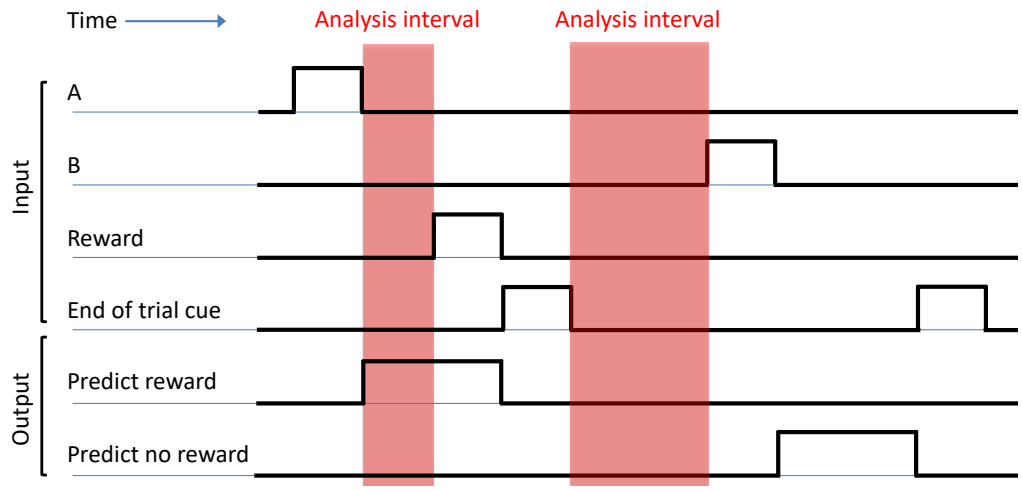

Fig. S6. RNN inputs and outputs for the context dependent trace conditioning task of Saez et al. (5). The RNN has four inputs (stimulus A, stimulus B, reward, and the end-of-trial cue) and two outputs (a reward predictive output and a no-reward predictive output). The mapping between stimulus and reward changes depending on context. During context 1 stimulus A is followed by a reward and stimulus B is not rewarded. During context 2 the associations are reversed and stimulus B is rewarded while stimulus A is not rewarded. The RNN's task is to predict the upcoming reward following the presentation of a stimulus by selecting the appropriate output. Outputs are shown for context 1. Highlighted in red are two intervals that we analyzed. On the left is the interval after stimulus offset and before the reward. On the right is the interval before stimulus onset.

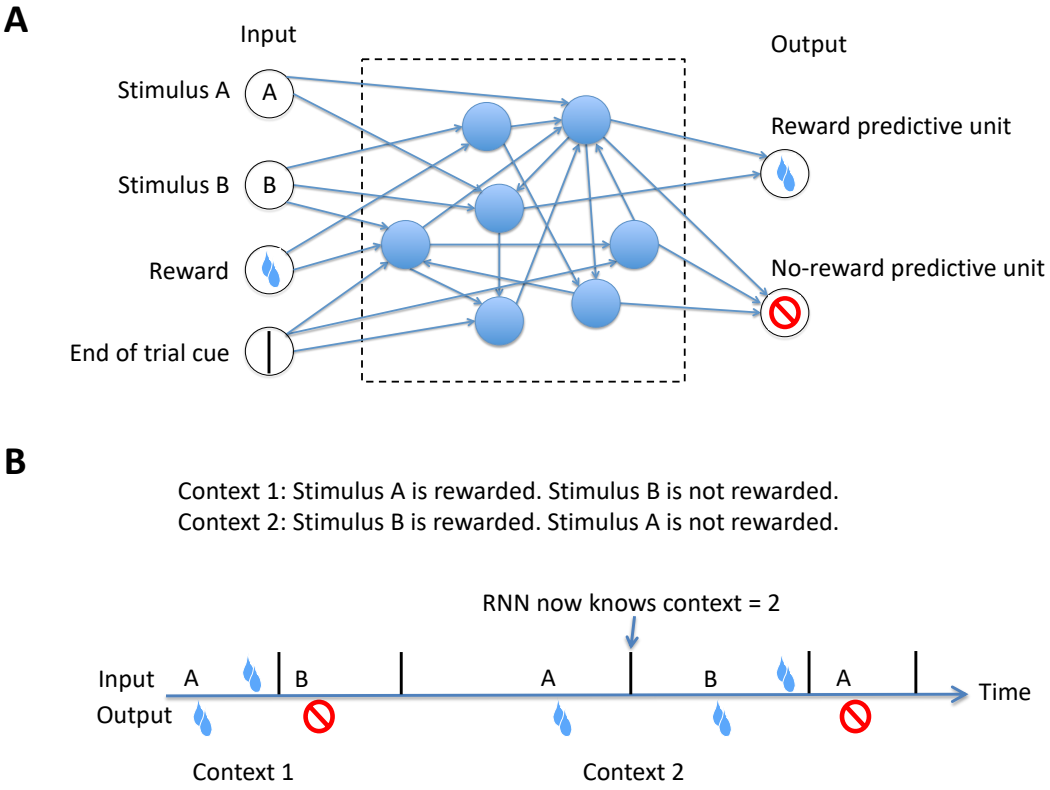

Fig. S7. RNN inputs and outputs for the context dependent trace conditioning task of Saez et al. (5) when the context changes. **(A)** The RNN has four inputs (stimulus A, stimulus B, reward, and the end-of-trial cue) and two outputs (a reward predictive output and a no-reward predictive output). The mapping between stimulus and reward changes depending on context. During context 1 stimulus A is followed by a reward and stimulus B is not rewarded. During context 2 the associations are reversed and stimulus B is rewarded while stimulus A is not rewarded. Note that a single pairing of stimulus/reward or stimulus/end-of-trial-cue is sufficient to determine the context. The RNN's task is to predict the upcoming reward following the presentation of a stimulus by selecting the appropriate output. **(B)** The inputs and outputs are shown for a single sequence with a change in context. Imagine a preceding stimulus/reward pairing (not shown) has established the context to be 1. Stimulus A and B are inputted and the RNN produces the correct outputs. The context now switches to 2. No explicit contextual cues are given to the RNN so when stimulus A is presented the RNN still responds with the appropriate output for context 1, by activating the reward predictive unit. No reward is inputted, as would be appropriate for context 1, so when the end-of-trial-cue appears the RNN now knows the context has changed to 2. For subsequent inputs the RNN now produces outputs appropriate for context 2.

The RNN for the ready-set-go interval reproduction task from Jazayeri and Shadlen (12) has two inputs (ready and set cues) and one output to indicate the interval between ready and set cues as shown in Figure S8. During training, the ready cue is followed by the set cue with an interval selected from a uniform random distribution between 200 and 1100 ms. The RNN output follows the set cue after a delay that matches the elapsed time between the ready and set cues. All inputs and outputs take the value of zero when they are "off" and one for a duration of 110 ms when they are "on". The RNN was trained on sequences of duration 4500 ms with multiple presentations of the ready and set cues in each sequence.

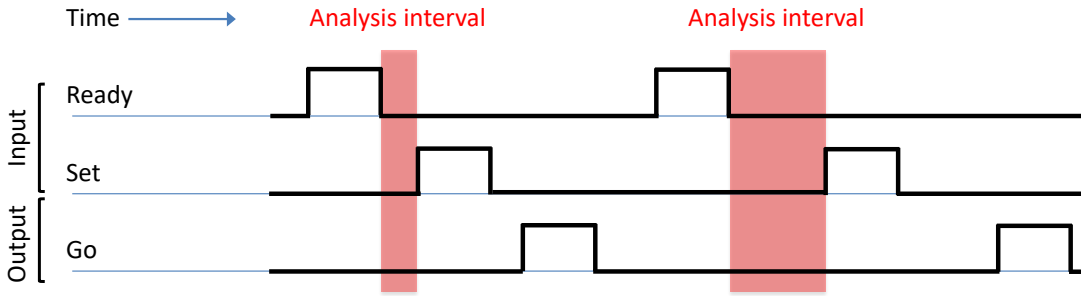

Fig. S8. RNN inputs and outputs for the ready-set-go interval reproduction task from Jazayeri and Shadlen (12). The RNN tracks the duration of the interval between ready and set cues (demarcated by two input pulses) in order to reproduce the same interval with a self initiated output at the appropriate time after the set cue. The interval we analyzed, between the ready and set cues, is highlighted in red.

The goal of the duration discrimination task from Genovesio et al. (13–15) is to compare the duration of two stimuli (S1 and S2) and select the stimulus that was presented for the longest duration. The RNN for this task has four inputs (S1, S2, go-cue for S1/S2 on left/right of screen, go-cue for S1/S2 on right/left of screen) and two outputs to indicate a hand response to either the right or left (Figure S9). The RNN was trained on sequences of length 5000 ms with a single presentation of S1, S2, and go-cue, per sequence. The order of events within a sequence is pre-stimulus period (uniformly distributed between 100 and 500 ms), S1 (uniformly distributed between 100 and 1500 ms), delay period (uniformly distributed between 0 and 1000 ms), S2 (uniformly distributed between 100 and 1500 ms), delay period (uniformly distributed between 0 and 1000 ms), and a go-cue that initiated the RNN output and remained on until until the end of the sequence. In the experiment, the go-cue was the presentation of the two stimuli (S1 and S2) simultaneously on the right and left side of the screen. On each trial the left and right assignment of S1 and S2 was random so the motor response could not be prepared in advance of this go-cue. To mimic this in the RNN we use two go-cues to indicate whether the position of S1 is on the right or left of the screen. The two RNN outputs then correspond to a hand response to either the right or left. The outputs are zero until the time of the go-cue, when the appropriate output becomes one until the end of the sequence.

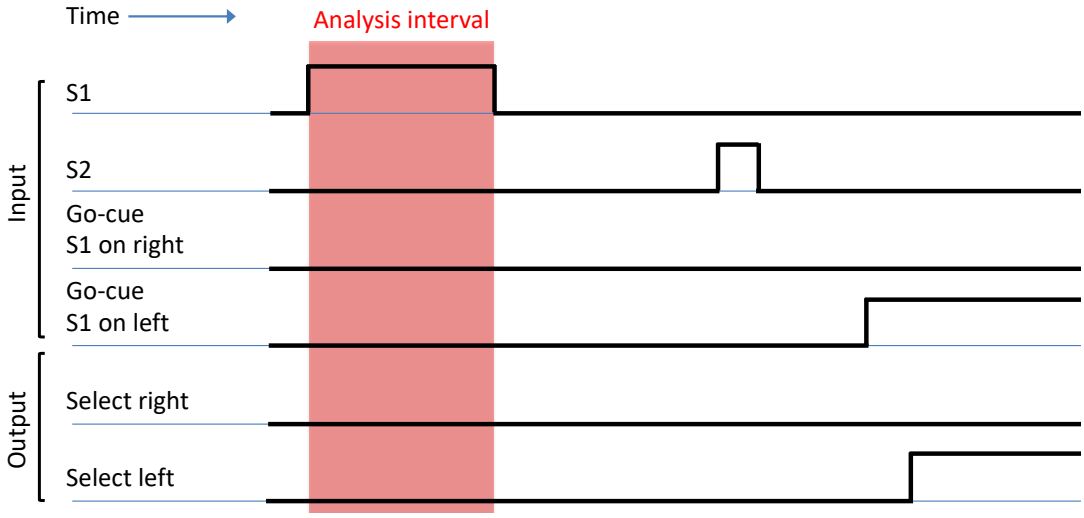

Fig. S9. RNN inputs and outputs for the duration discrimination task of Genovesio et al. (13). The RNN compares the duration of two stimuli (S1 and S2) and then reports which stimulus was on longer. The interval we analyzed is highlighted in red.

A. Stabilized random RNN. We analyzed the pretrained network accompanying the paper of Laje and Buonomano(6). To ensure the number of units is similar across datasets, we randomly selected 100 out of the 800 units in the network for further analyses. We generated and analyzed 100 trials from the network by adding Gaussian random noise, with zero mean and standard deviation of 0.2, at each timestep of the simulation. The firing rate of each of the 100 units is stored every 100 ms and this activity is used in all subsequent analyses.

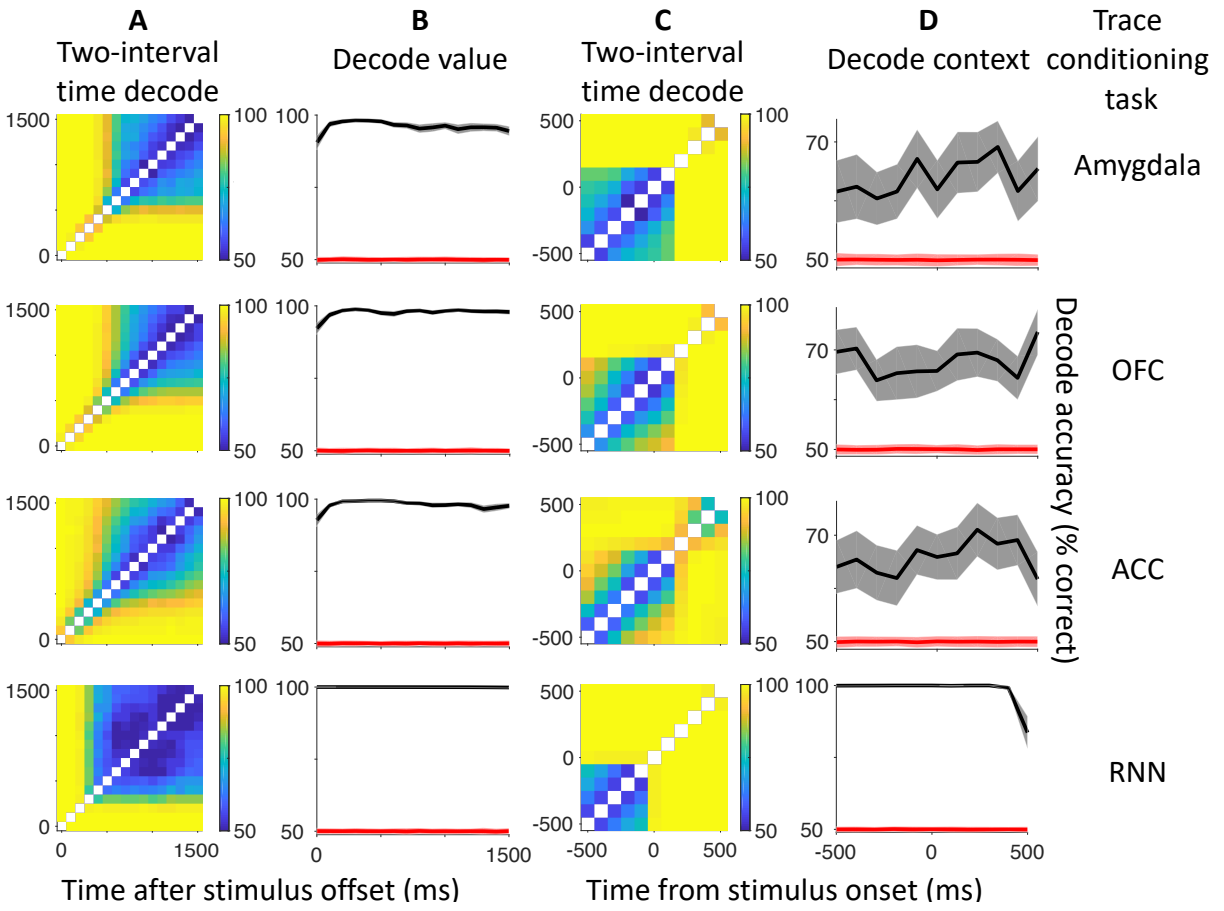

Fig. S10. Decoding time from neural activity reveals signatures of fixed point dynamics. **(A, C)** Pixel(i,j) is the decode accuracy of a binary classifier trained to discriminate timepoints i and j using 100 ms bins of neural activity. The blocks of time where the decode is near chance level (50%) are signatures of fixed point dynamics. However, there is the possibility the dynamics evolve on longer timescales than we can observe with the delay intervals used in this experiment. The pattern of fixed points seen in the data agree with the RNN model. In the model, the fixed points before stimulus onset store contextual information (column C) and the fixed points after stimulus offset encode the expected reward and the stimulus (column A). **(B, D)** Importantly, a linear classifier can easily discriminate other task relevant quantities during these time intervals so the poor time-decode is not simply due to noisy neural responses that have lost all informational content. The black curve shows the decode accuracy of a binary classifier trained to discriminate reward and no-reward trials (B) and trials from context 1 and context 2 (D) using 100 ms bins of neural activity. Error bars show two standard deviations.

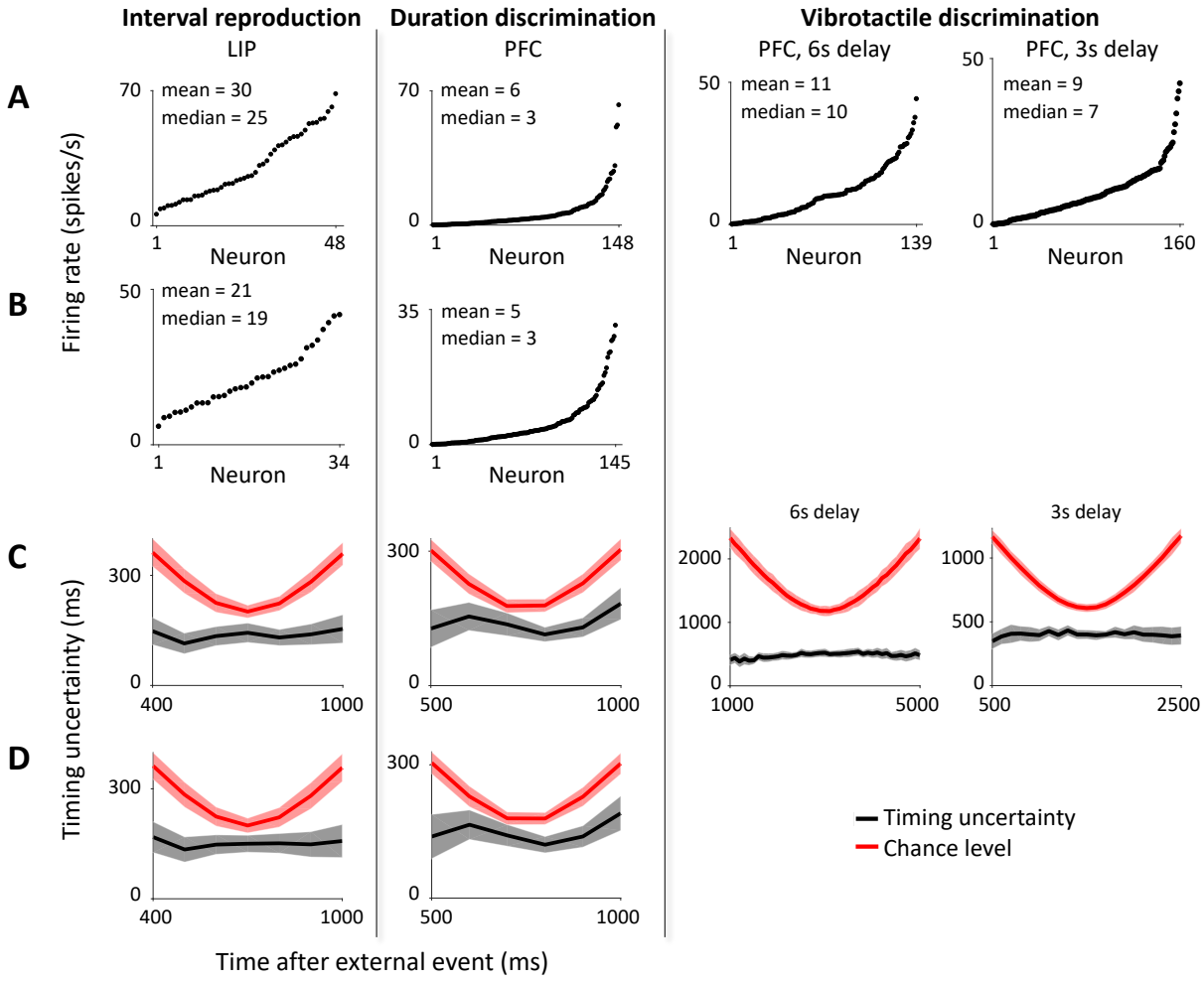

Fig. S11. The timing uncertainty of the interval reproduction task (left column) and duration discrimination task (second column from the left) does not depend on the highest firing neurons. Furthermore, the differences between the timing uncertainty of these two tasks where tracking time is explicitly required and the vibrotactile discrimination task (two columns on the right) does not depend on the highest firing neurons. The intervals considered here are the same as in Figures 5 and 6. **(A)** The mean firing rate across time and trials for each neuron. Neurons are sorted according to their mean firing rate. Each neuron is recorded for a mean/median number of trials of 60/49 (interval reproduction), 59/56 (duration discrimination), 91/91 (vibrotactile discrimination, 6s delay), and 159/176 (vibrotactile discrimination, 3s delay). The data summarized in row (A) is used to generate Figure 5 and also row (C). **(C)** The timing uncertainty results shown in Figure 5 are reproduced here for ease of comparison. **(B)** The smaller timing uncertainty seen in the interval reproduction and duration discrimination tasks versus the vibrotactile discrimination task is not due to a few higher firing neurons. To verify this, we note that the maximum mean firing rate of neurons in the vibrotactile discrimination task during the intervals shown in Figure 5 is 42 Hz, so we remove any neuron with mean firing rate over 42 Hz. The resulting distribution of mean firing rates across time and trials for each neuron is shown in (B). Neurons are sorted according to their mean firing rate. Each neuron is recorded for a mean/median number of trials of 51/43 (interval reproduction) and 59/56 (duration discrimination). The data summarized in row (B) is used to generate the timing uncertainty shown in (D). **(D)** The timing uncertainty when the highest firing neurons are removed (row D) is similar to the original timing uncertainty when all neurons are included (row C). The timing uncertainty in (D) is smaller than the timing uncertainty for the vibrotactile discrimination task so the differences are not the result of a few higher firing neurons, as we have removed these high firing neurons before generating (D). Furthermore, the larger timing uncertainty in the vibrotactile discrimination task is not due to smaller mean or median firing rates in this task. The mean and median firing rates of neurons are smaller for the duration discrimination task (mean/median = 6/3 Hz with all neurons, mean/median = 5/3 Hz after removing highest firing neurons) than the vibrotactile discrimination task (mean/median = 11/10 Hz for 6s delay, mean/median = 9/7 Hz for 3s delay).

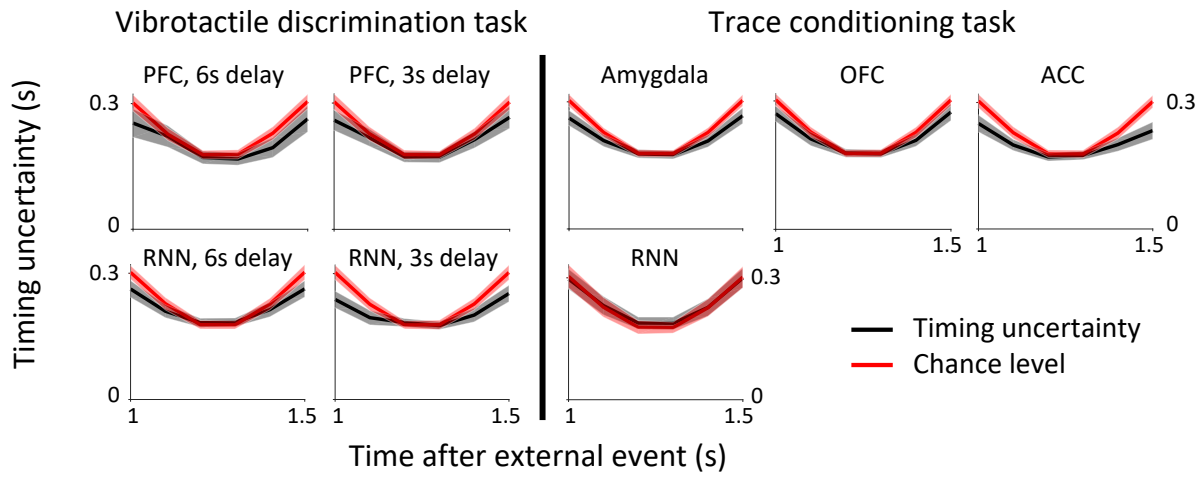

Fig. S12. The timing uncertainty is consistent for both the vibrotactile discrimination and trace conditioning tasks. The slowly varying dynamics observed in the vibrotactile discrimination task are consistent with the putative fixed point dynamics observed in the trace conditioning task. The timing uncertainty is shown during the 500 ms interval with putative fixed point dynamics in the trace conditioning task, and for an arbitrary 500 ms interval after the transient following stimulus offset in the vibrotactile discrimination task. In all brain regions the timing uncertainty is near chance level. Longer time-scale fluctuations may not be as apparent in the trace conditioning task because a shorter delay period was used in the experiment, preventing neural dynamics from evolving sufficiently in the absence of external stimuli. Error bars show two standard deviations.

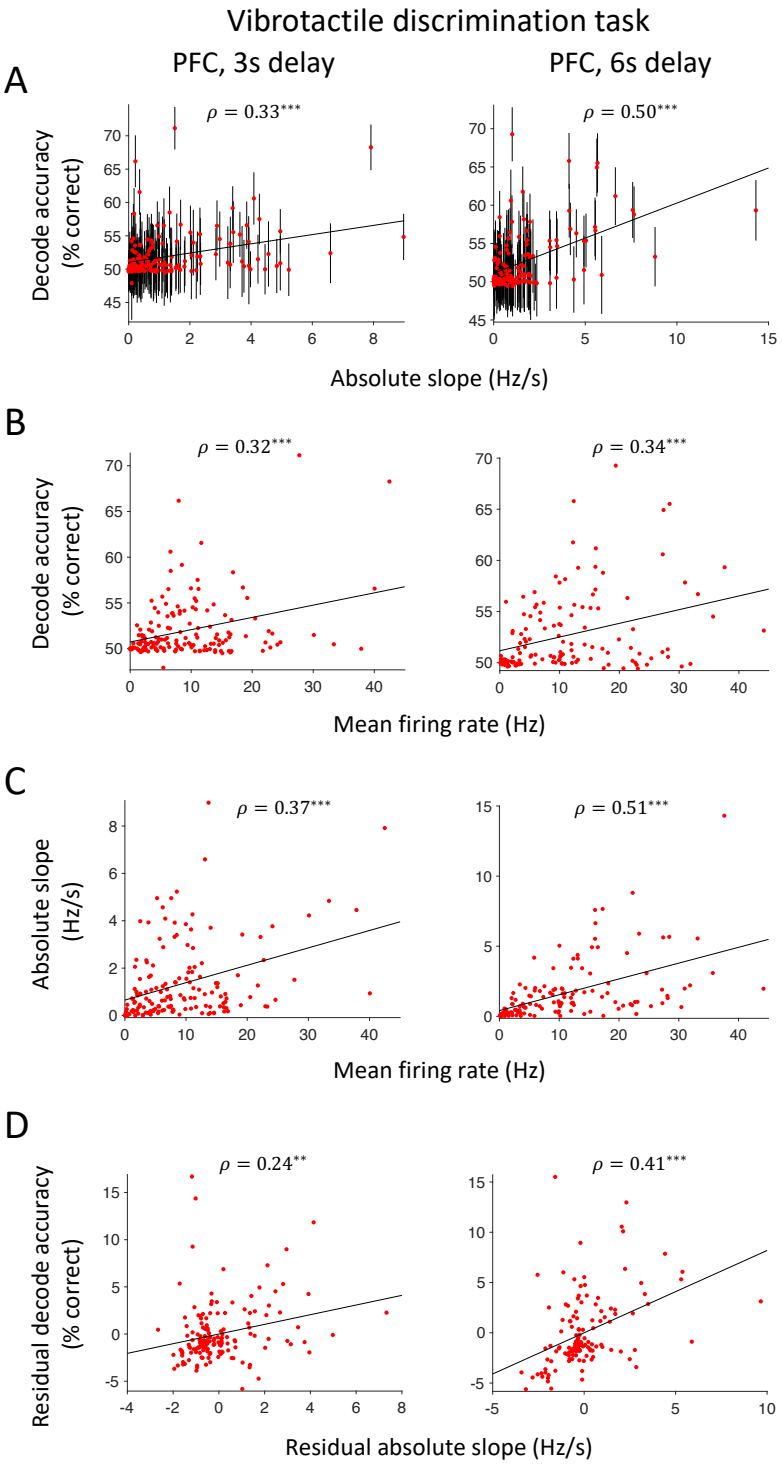

Fig. S13. Neurons that are more informative about time (i.e. those with ramping activity with a larger absolute slope) are also more informative about task relevant information (i.e. the frequency of the vibrotactile stimulus). There is not one population that encodes time and a separate population that encodes task relevant information. The intervals are the same as in Figures 5 and 6: 500 to 2500 ms after stimulus offset for the three second delay period and 1000 to 5000 ms after stimulus offset for the six second delay period. Pearson's correlation values ρ are reported (*p < 0.05, **p < 0.01, ***p < 0.001) and linear regression fits are shown (black lines). **(A)** The scatterplot shows one set of metrics for how well individual neurons encode task relevant information (decode accuracy for vibrotactile frequency) and time (absolute slope of the linear fit to firing rate). Each dot corresponds to one neuron. The decode accuracy is from a binary classifier trained to discriminate low versus high values of the vibrotactile frequency. Error bars show two standard deviations. The correlation between decode accuracy and absolute slope would be negative in the case of two separate specialized populations. Instead we observe that it is positive. This positive correlation is only partially explained by the firing rate of neurons (most active neurons tend to be important for encoding both variables). Indeed, neurons that have higher firing rates are more informative about the vibrotactile frequency **(B)** and neurons that have higher firing rates are more informative about time, i.e. having larger absolute slopes **(C)**. **(D)** Same as (A) but the component due to cell activity was removed from the data. Specifically, the y-value is the decode accuracy from panel (A) minus the linear regression fit from panel (B). The x-value is the absolute slope from panel (A) minus the linear regression fit from panel (C). Residuals also show a positive correlation.

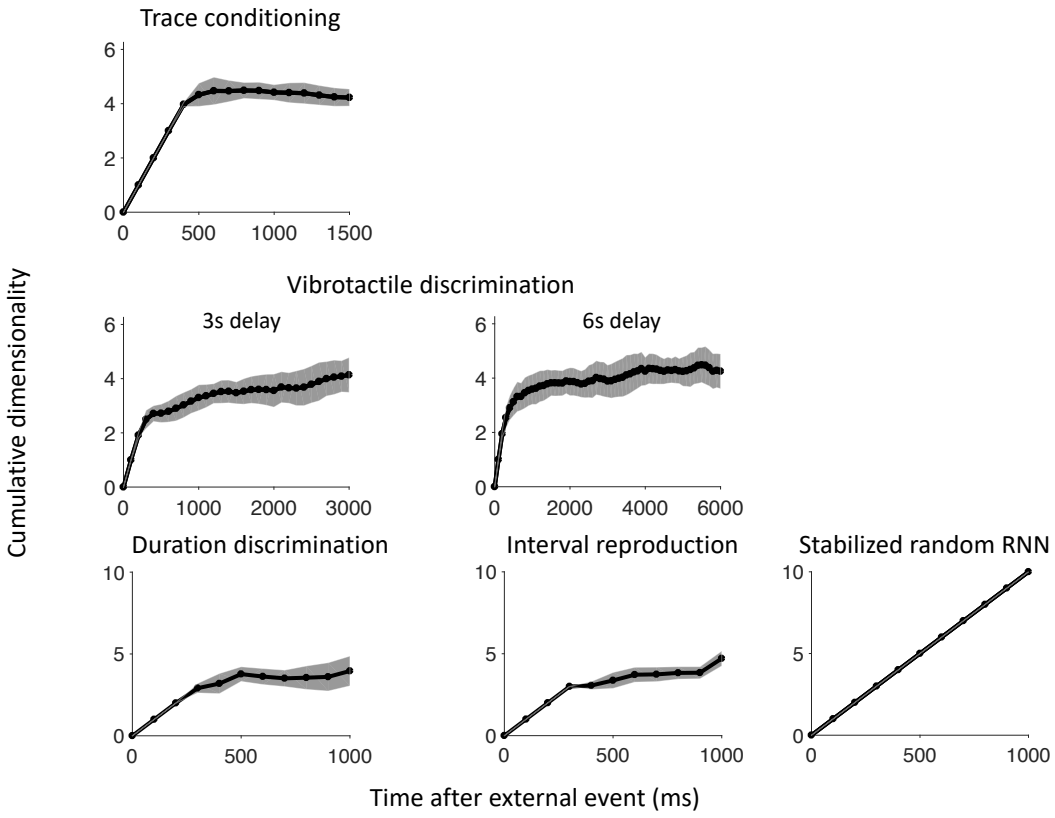

Fig. S14. Cumulative dimensionality for the RNN models. Error bars show one standard deviation.

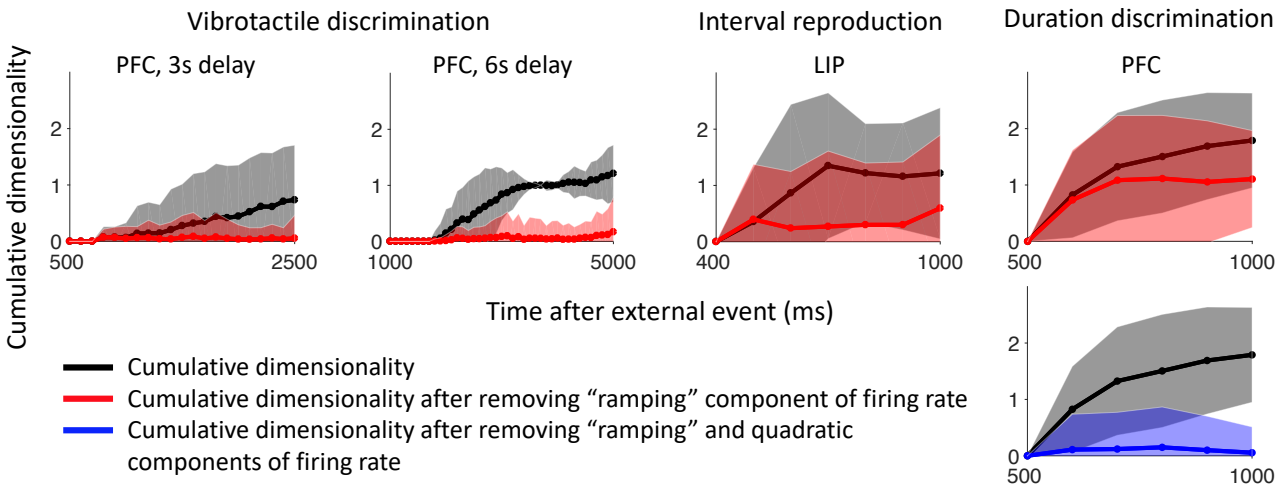

Fig. S15. Cumulative dimensionality of the original data (black curves), after removing the linear "ramping" component of the firing rate (red curves), and after removing both the linear and quadratic components of the firing rate (blue curve). Time intervals are the same as in Figures 5 and 6, but different than the intervals in Figure 7. After removing the ramping component of the neural activity the cumulative dimensionality decreases for all datasets. Notice the cumulative dimensionality does not always drop to zero. This is because not all of the consistent trial-to-trial activity is captured by linear ramping dynamics. For example, in the duration discrimination task the cumulative dimensionality is also influenced by the quadratic component of each neuron's firing rate. For each neuron, we fit the mean firing rate with a quadratic function, and then subtract this fit from the single trial firing rates before computing the cumulative dimensionality to obtain the blue curve that saturates near zero. Error bars show two standard deviations.

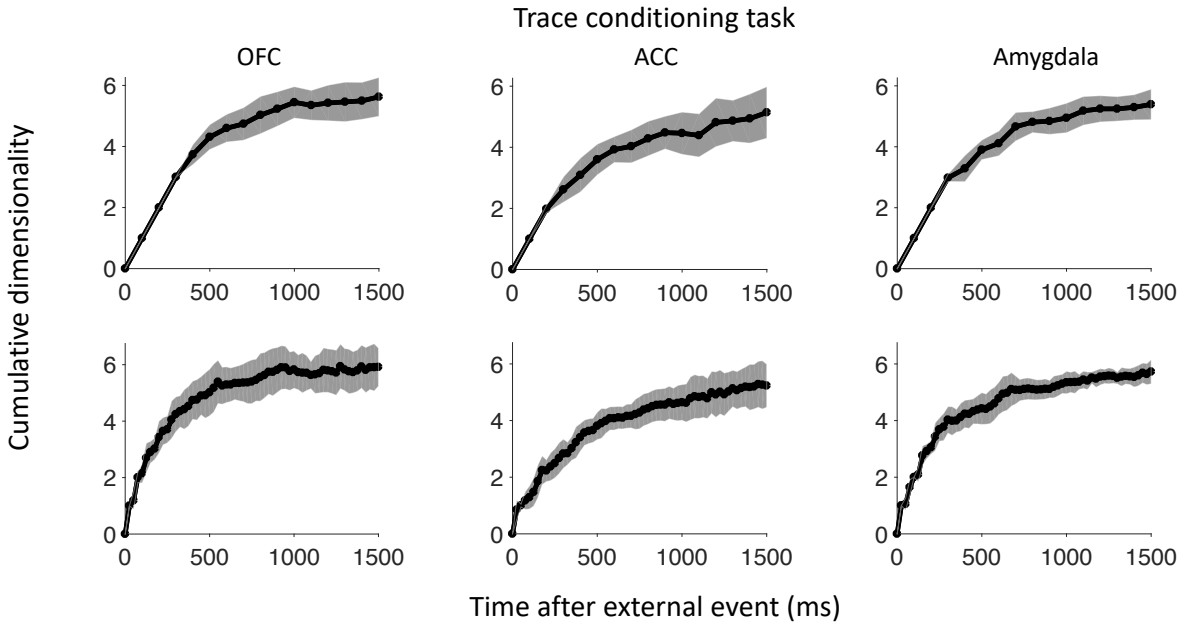

Fig. S16. The cumulative dimensionality for the trace conditioning task (5) is stable as the number of timepoints along the neural trajectory are varied. In the top row, firing rates are from nonoverlapping 100 ms bins calculated every 100 ms. In the bottom row, firing rates are from overlapping 100 ms bins calculated every 25 ms. Error bars show one standard deviation.

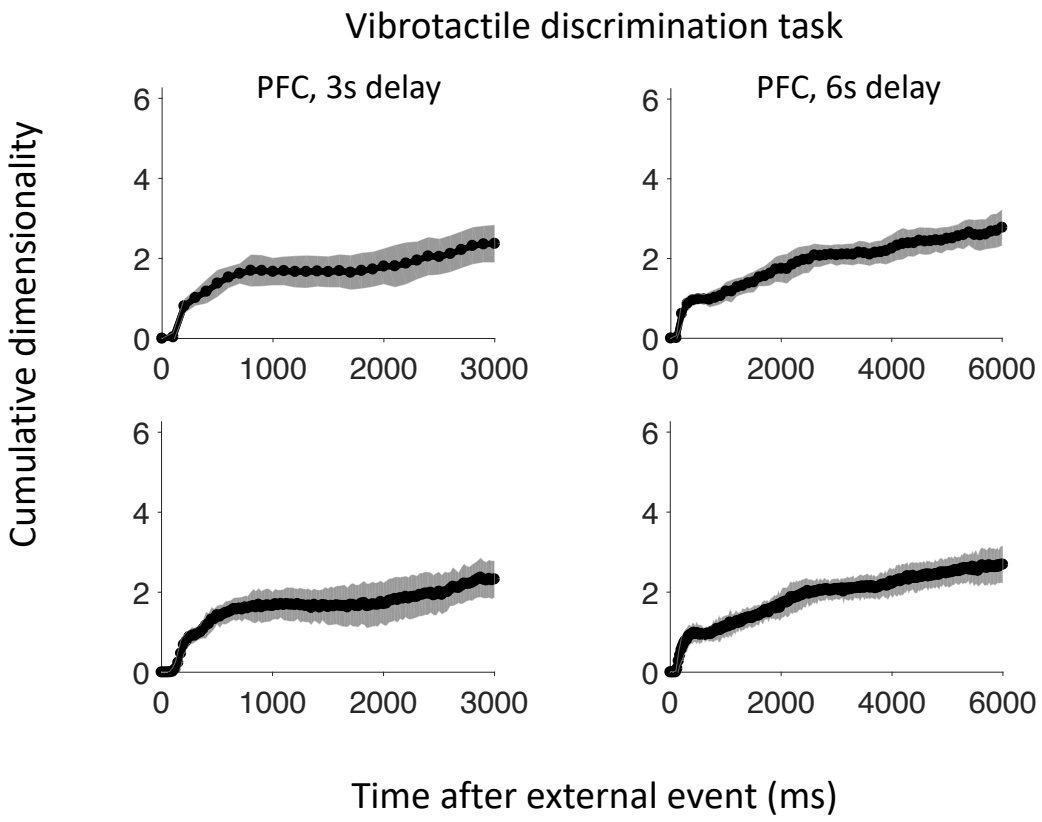

Fig. S17. The cumulative dimensionality for the vibrotactile discrimination task (1) is stable as the number of timepoints along the neural trajectory are varied. In the top row, firing rates are from nonoverlapping 100 ms bins calculated every 100 ms. In the bottom row, firing rates are from overlapping 100 ms bins calculated every 25 ms. Error bars show one standard deviation.

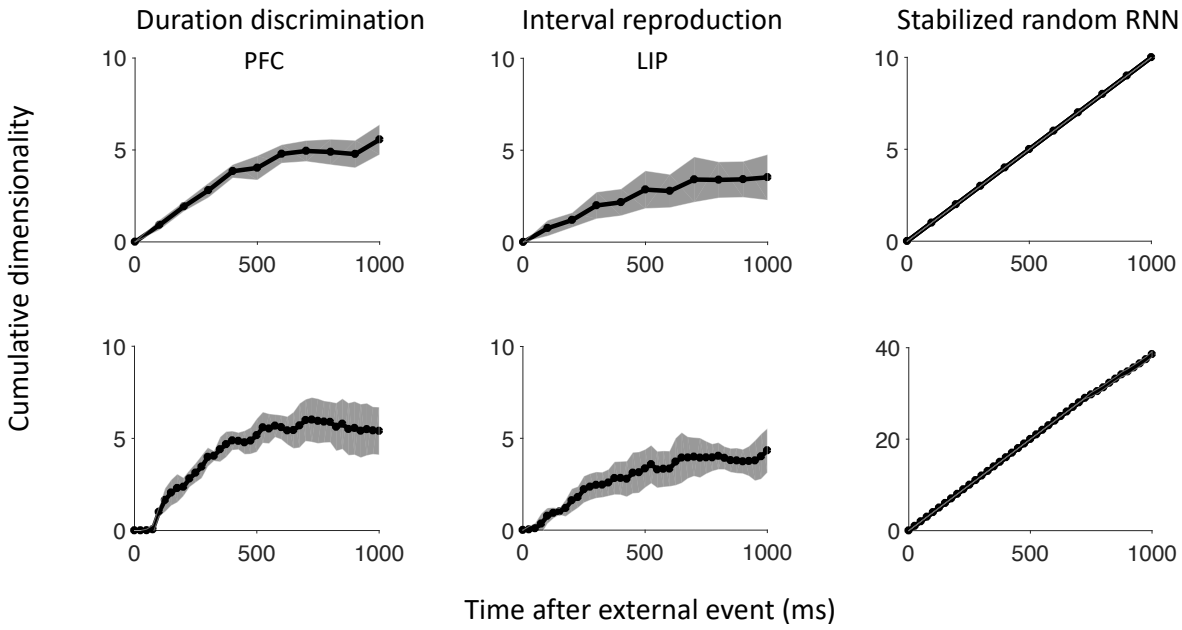

Fig. S18. For the duration discrimination task (13) and interval reproduction task (12), the cumulative dimensionality is stable as the number of timepoints along the neural trajectory are varied. In the top row, firing rates are from nonoverlapping 100 ms bins calculated every 100 ms. In the bottom row, firing rates are from overlapping 100 ms bins calculated every 25 ms. For the stabilized random RNN of Laje and Buonomano (6), the cumulative dimensionality grows as the number of timepoints are sampled more densely. Error bars show one standard deviation.

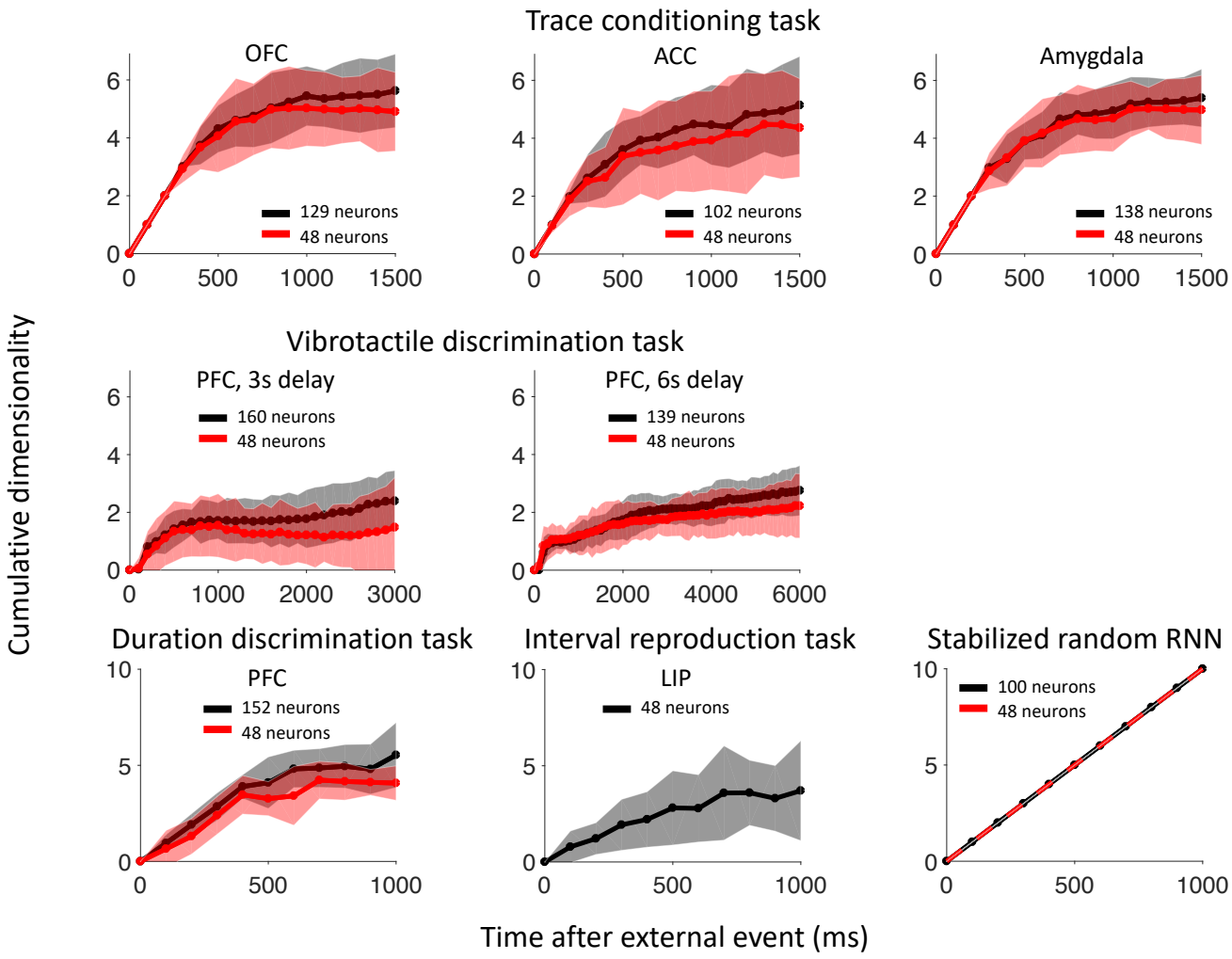

Fig. S19. The cumulative dimensionality is stable when the number of neurons is equalized between datasets. The interval reproduction dataset from Jazayeri et al. (12) had the fewest neurons (48) that met our criteria for duration of delay interval and number of trials so to compare datasets we randomly selected 48 neurons from the other datasets and recomputed the cumulative dimensionality. Black curves show the cumulative dimensionality with all neurons (same as Figure 7) and red curves show the cumulative dimensionality with 48 randomly selected neurons. Error bars show two standard deviations.

References

1. R Romo, CD Brody, A Hernandez, L Lemus, Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature* (1999).
2. LF Abbott, K Rajan, H Sompolinsky, *Interactions between Intrinsic and Stimulus-Evoked Activity in Recurrent Neural Networks.* (Oxford university press), (2011).
3. L Mazzucato, A Fontanini, G La Camera, Stimuli reduce the dimensionality of cortical activity. *Front. Syst. Neurosci.* (2016).
4. P Gao, et al., A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv* (2017).
5. A Saez, M Rigotti, S Ostojic, S Fusi, C Salzman, Abstract context representations in primate amygdala and prefrontal cortex. *Neuron* (2015).
6. R Laje, DV Buonomano, Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nat. neuroscience* (2013).
7. V Mante, D Sussillo, KV Shenoy, WT Newsome, Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* (2013).
8. D Sussillo, MM Churchland, MT Kaufman, KV Shenoy, A neural network that finds a naturalistic solution for the production of muscle activity. *Nat. neuroscience* **18**, 1025–1033 (2015).
9. J Martens, I Sutskever, Learning recurrent neural networks with hessian-free optimization. *ICML* (2011).
10. AM Saxe, JL McClelland, S Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120* (2013).
11. O Barak, D Sussillo, R Romo, M Tsodyks, L Abbott, From fixed points to chaos: three models of delayed discrimination. *Prog. neurobiology* (2013).
12. M Jazayeri, MN Shadlen, A neural mechanism for sensing and reproducing a time interval. *Curr. Biol.* (2015).
13. A Genovesio, S Tsujimoto, SP Wise, Feature- and order-based timing representations in the frontal cortex. *Neuron* (2009).
14. A Genovesio, R Cirillo, S Tsujimoto, SM Abdellatif, SP Wise, Automatic comparison of stimulus durations in the primate prefrontal cortex: the neural basis of across-task interference. *J Neurophysiol* (2015).
15. E Marcos, S Tsujimoto, A Genovesio, Independent coding of absolute duration and distance magnitudes in the prefrontal cortex. *J Neurophysiol* (2017).