Continuous Attractor Networks with Realistic Neural Dynamics

T.C. Wyllie

St Catharine's College

June 2021

Supervisor: Prof. M. Lengyel
Department of Engineering
University of Cambridge

I hereby declare that, except where specifically indicated, the work submitted herein is my own original work.

Signature: [illegible]

Date: 2nd June 2021

Acknowledgements

I would like to express my gratitude to Máté Lengyel and the Computational and Biological Learning Lab for assisting me in undertaking this thesis. I would also like to pay particular thanks to Jake Stroud for all his time spent providing me with helpful feedback and interesting suggestions for directions to take this project in.

Technical Abstract

Working memory is essential for cognitive functions such as problem solving and planning, and it has been proposed that persistent activity in neural circuits underlies working memory. Attractor neural networks can use strongly recurrent connections to form persistent states, and information is encoded in these networks as locations in neural state-space. In particular, continuous variables can be encoded as locations on continuous manifolds of attractors in neural state-space. Existing models of these attractor networks (for example, bump attractors) are hand-crafted, difficult to tune, and are based around symmetric connectivity matrices which significantly restrict their dynamics. Specifically, current models are unable to account for dynamic coding that is observed in monkeys during the delay period of a spatial working memory task (memory guided saccade); instead they predict a stable code, i.e. that the memory is stored as a single point in state-space for the duration of working memory maintenance.

In this research we investigate recurrent neural network models that we have trained to solve a generic working memory maintenance task. Importantly, we do not require any symmetry from our networks. We train these networks using a cost function that we have devised, that only requires the network to be able to maintain information for a 1.5 second delay period.

We implement these recurrent neural networks and the cost function in TensorFlow in Python, and show that our trained networks display attractor dynamics using a fixed point analysis. We also visualise both the network dynamics and the attractor manifold using principal component analysis. We finally undertake a decodability analysis, computing the full cross-temporal decodability, and also the alignment index, to determine that our networks exhibit a dynamic code.

Our key conclusions are, firstly, that it is possible to create continuous attractor networks without strict requirements on the connectivity matrix. Secondly, continuous ring attractor manifolds can arise even if the network is only required to store a discrete range of inputs. Finally, and most importantly, we demonstrate that the asymmetry of our networks allows them to exhibit dynamic coding.

Contents

1 Introduction

Any useful memory system must have the ability to store information that persists over time and can subsequently be recollected. The duration for which memories persist in the brain varies from seconds (short-term memory) [1] to years and even decades (long-term memory) [2]. Short-term memory is a key aspect of working memory, which is short-term storage and manipulation of information for later use in reasoning and decision making [3]. Working memory is essential for cognitive functions such as problem solving and planning [3], [4], so the neural mechanisms underpinning working memory are of great importance.

Many different biological processes have temporal persistence that could reasonably underlie memory in the brain, and these occur over timescales varying by orders of magnitudes. Structural changes in the brain, i.e. the varying of synaptic strengths and connections, are understood to be sufficiently persistent that they facilitate storage for long-term memory [1], [5]. However, the changes required to 'write' these long-term memories occur over the course of hours to days [6], [7], which is clearly too slow to explain working memory. So what are the mechanisms of working memory? Individual neurons can respond quickly to changes, for example ion concentration time constants are on the order of milliseconds to tens of milliseconds [8] and postsynaptic potentials last for tens to hundreds of milliseconds [9], but the timescales of these biochemical state variables are not long enough to provide a direct explanation of working memory [10]. Additionally, neural activity is highly noisy [1] and so simple representations of information using individual state variables are implausible.

In their classic 1989 paper, Funahashi et al. [11] demonstrated that persistent network activity could be the basis of working memory. They trained monkeys to perform a memory guided saccade task, which is a spatial working memory task. The memory guided saccade task (figure 1) is as follows: a visual cue briefly appears in one of several locations; the cue disappears and there is a 'delay period' of no cue being displayed; the subject must recall which one of the original locations the cue appeared in by enacting a saccade towards it. This task requires the subject to store the cue location in their working memory for the duration of the delay period. Funahashi et al. recorded prefrontal cortex (PFC) neurons in the monkeys which displayed sustained elevated firing rates when the maintenance of a working memory was required. Importantly, the increases in firing rate were selective to the information that the monkey was tasked with remembering, suggesting that the persistent activity of neurons meaningfully encodes information. Such persistent neuron activity is facilitated by neural circuits with strongly recurrent excitatory connections [12]–[14], as the presence of recurrent connections allows for positive feedback which enables the creation of stable, persistent 'attractor' states [1], [15]. The modelling of recurrent neural networks with persistent activity has become a ubiquitous framework for understanding short-term memory [10], [16], [17] and working memory [12], [18]–[20].

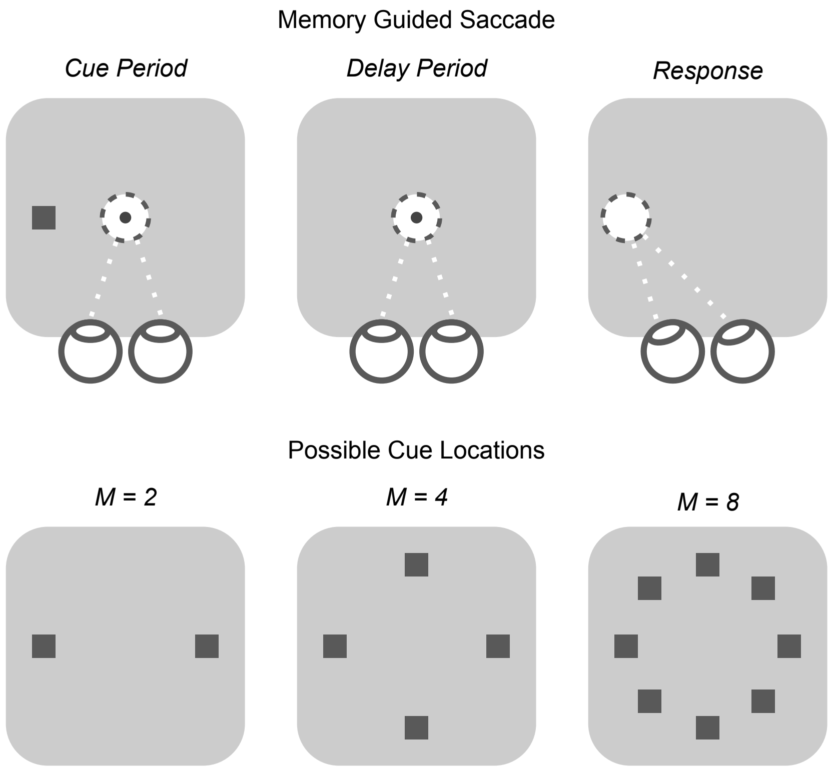

Figure 1: Upper - memory guided saccade task. The black square represents the cue location. Lower - the cue location is randomly chosen from a set of M possible locations around a circle.

1.1 Attractor Neural Networks

Attractor neural networks are some of the most extensively studied models of neural circuits. It has been proposed variously that attractor dynamics underlie neural activity in: head direction circuits [21], [22]; the oculomotor system [23]; the hippocampus [24]–[27]; the prefrontal cortex [17], [19]. The defining feature of an attractor network is the ability to maintain persistent levels of neural activity after the removal of an input [1]. The 'state' or 'activity' of the network is given by the activity (firing rate) of each individual neuron, and information may then be encoded by a location in neural state-space [28]. States in the proximity of an attractor state will be drawn into the attractor state, and if reached the network will remain in the attractor state indefinitely. Attractor states can either be separate, individual points in the state-space of a network, or exist as a continuum, such as a line or a ring [25], [29], [30]. Attractor networks are accordingly either 'discrete' or 'continuous', and both are relevant as models of neural circuits [28], [31], [32]. Unsurprisingly, discrete attractor networks are generally associated with the storage of discrete i.e. categorical information [33], [34] whereas continuous attractor networks are more suitable for storing continuous variables, such as the angle around the circle of a cue in the memory guided saccade task [28]. We often refer to these networks simply by the topology of their attractor manifolds, e.g. 'ring attractors' or 'line attractors'. Neural dynamics, i.e. changes in activity over time, can be represented as paths in state-space; if regions of state-space cause these paths to vary very slowly then these regions are effectively attractors. These are also called 'slow regions' of state-space.

1.2 Attractor Networks for Working Memory

Attractor networks have proved to be fruitful models of working memory [17], [20], [25], [35]. One such model for encoding cyclic variables such as direction [36] or hue [37] is the 'bump attractor' [19], [20], [35]. The bump attractor is a ring attractor where neurons that are selective to different values of the input form a bump of activity along the ring, i.e. the value encoded by the network is the location of a bump (figure 2). The network activity corresponding to this bump is persistent over time [19] (figure 2) - which is the key requirement for any useful memory model. Besides this fundamental property bump attractor models have shown promise in explaining other neural phenomena [19], [38] such as the increase in imprecision of working memory over time (proposed by [19] to be the result of noise causing drift along the ring, shown in figure 2). But despite these successes, current continuous attractor network models [19], [35], [39] have some notable inadequacies.

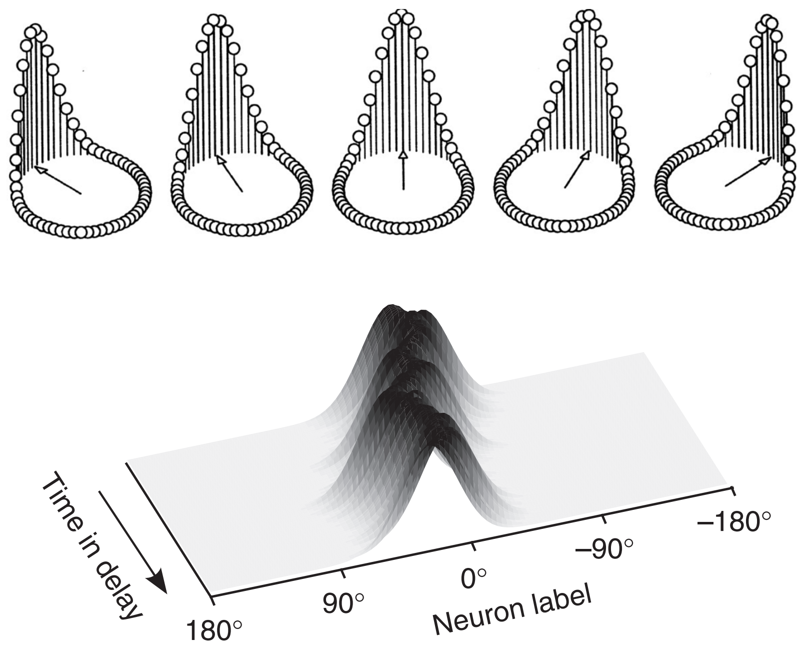

Figure 2: Upper - the bump attractor model encodes a circular variable (shown by the arrow) as the location of a bump on a ring. Adapted from [40]. Lower - bump activity persists over time (but drifts slowly). Adapted from [19].

The first problem with current attractor neural network models, in simple terms, is that their structure is unrealistic. In 1995, Ben-Yishai et al. [30] presented their ring attractor model of orientation tuning. This model is constructed by carefully choosing network connectivity weights only as a function of distance between the angles for which neurons are most selective. Current models [19], [35], [39] use similar methods to hand-craft continuous attractor networks, i.e. use connectivity that is effectively of the form $W_{ij} = f(|i - j|)$ - which creates symmetric connectivity matrices. These networks depend on this symmetry to realise their attractor dynamics [29], [30], [41], but biology is inherently heterogeneous and synaptic connections in the brain are not expected to obey such rigid symmetry [39], [42]. Furthermore, symmetric matrices are diagonalisable, which essentially negates any interconnectedness of the network. This problem is evident when analysing the tuning curves of neurons within a circuit. Ring attractor models based on symmetric connectivity predict homogeneous tuning curves, where every neuron shows the same response (up to a phase shift) to the input condition. However, real measurements of tuning curves show heterogeneity [42]. An additional problem with these models is that recipes to hand-craft connectivity matrices which form continuous attractor networks are difficult to determine and difficult to tune [39].

1.3 Stable and Dynamic Coding

The second problem with current models (section 1.2) that we address is their inability to produce certain aspects of neural dynamics that are observed in the brain. To understand this issue we must first distinguish between 'stable coding' and 'dynamic coding'.

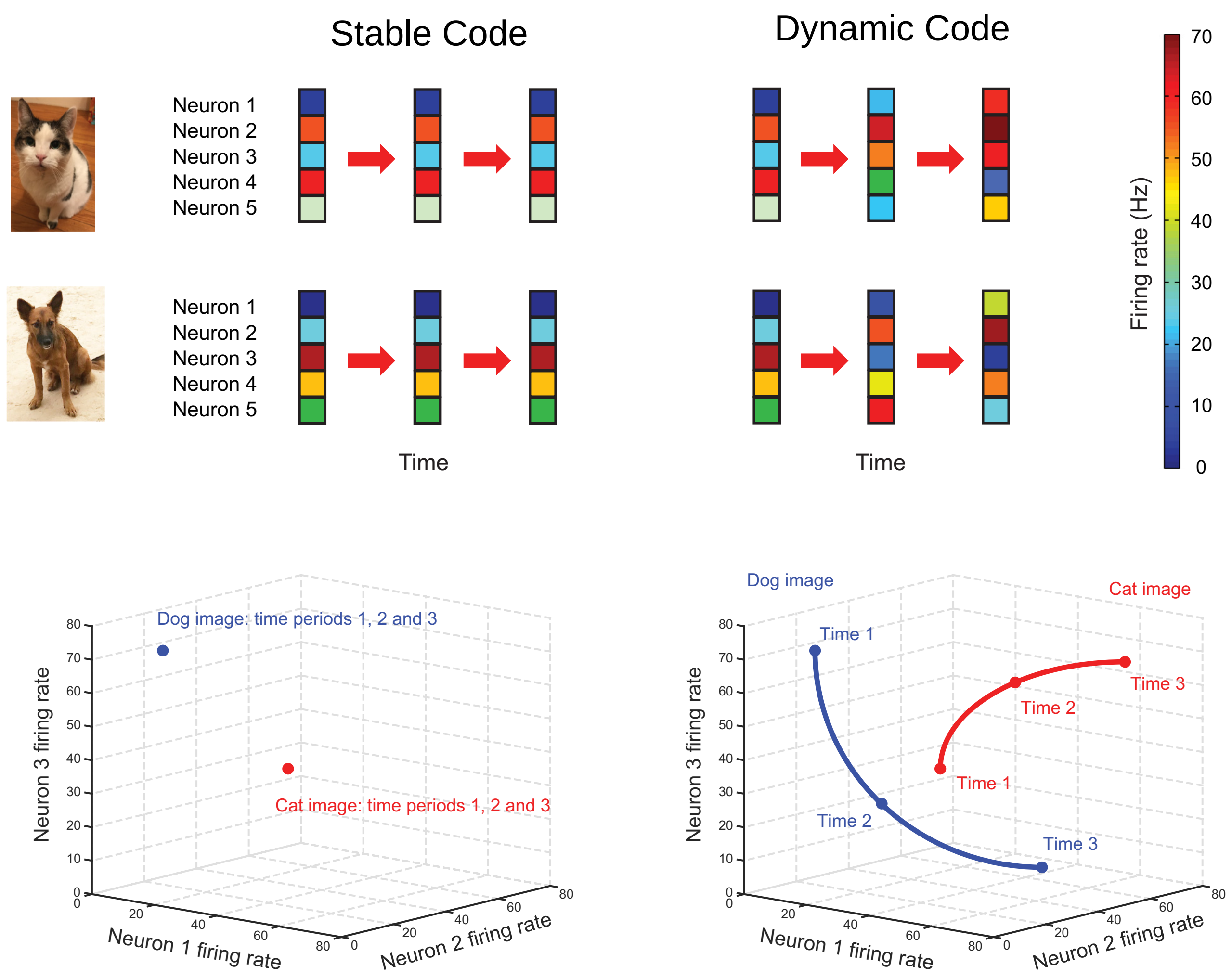

Figure 3: Representation of how network activity could encode categorical information in neural state-space, and how this coding may be either stable or dynamic. Figure adapted from [43].

Persistent activity has been proposed as the basis for working memory [43], but even if this was a known certainty, it still leaves open the question of *how* neural circuits with persistent activity actually *use* persistent activity to encode information. In other words, what is the mapping between the neural activity in a neural circuit, and the information (memory) that is encoded by that circuit?

Perhaps the simplest answer to this coding question is that there is a one-to-one mapping between points in a subset of neural state-space and the set of information that the network can encode. This implies that anytime the circuit is required to store that piece of information, it enters the corresponding state. This is demonstrated for a 2-class categorical problem on the left of figure 3, where one state corresponds to a dog, and the other to a cat. This is 'stable coding' [43]. In contrast, 'dynamic coding' is a many-to-one mapping from neural states to encoded information, where the network traverses a state-space trajectory during memory maintenance [43]–[45]. Figure 3 shows how dynamic coding might be used to encode two classes with arbitrary state-space trajectories.

Bump attractor networks use an inherently stable coding - the network 'remembers' its exact input state and persists in that state for the duration of the delay period of the task. Figure 2 (lower) shows this clearly, the bump moves forward in time, but is effectively the same bump - and so the same coding - for all time in the delay period. However, there is evidence to suggest that the coding used by neural circuits involved with working memory is in fact highly dynamic, particularly during the first few hundred milliseconds after cue presentation [45]–[49].

For example, Spaak et al. [45] trained monkeys to perform a memory guided saccade, and recorded the activity of PFC neurons. To assess evidence of dynamic coding they trained classifiers to discriminate between the cue locations the monkey had received (and was storing in working memory in order to complete the task), based on temporal slices of the recorded neural activity. Training a classifier in this fashion effectively learns the coding that the network is using at that time of the task. By assessing the performance of a classifier (i.e., a code) trained using one time slice of the task, on another time slice, it can be determined how well the code generalises across the different times of the task. For a stable code, the generalisation should be strong between all pairs of time points, but with dynamic codes, points at very different times will use very different codes and so exhibit low decodability. Figure 4 shows the full cross-temporal decodability analysis on these recordings, and clearly exhibits a strongly dynamic code at the start of the task.

So why can't we just 'fix' existing continuous attractor network models to demonstrate dynamic coding? As discussed in section 1.2, constructing a continuous attractor network is not a trivial undertaking [35], [39], and even with the significantly restrictive constraint of connectivity matrices being symmetric they are difficult to tune, requiring careful setting of parameters [1], [30]. No continuous attractor network models displaying dynamic coding have yet been established in the literature.

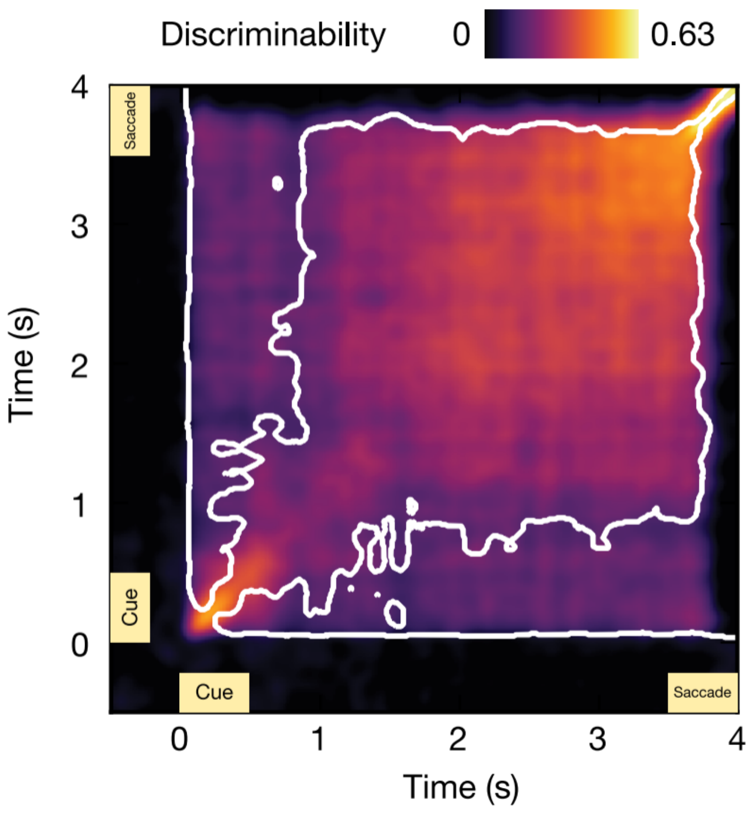

Figure 4: Cross-temporal decodability (discriminability) of electrophysiological recording data from monkeys during a memory guided saccade task. The white contour line shows that the decoding determined using early activity generalises poorly to late times, indicating strongly dynamic coding. Figure adapted from [45].

1.4 Can Continuous Attractor Models Exhibit Dynamic Coding?

In this work we study Recurrent Neural Network (RNN) models capable of performing working memory maintenance. We present a cost function with which to train models on to solve a working memory task. We then analyse the properties of the trained networks to demonstrate that they exhibit attractor dynamics. We explore how training on a task with a discrete number of visual cue orientations can generate a network capable of continuous encoding of the visual cue, if sufficiently many discrete angles are used.

We then demonstrate how constraining a network to be symmetric (in the sense that its network weights are symmetric) during training restricts the complexity of the resulting dynamics; in contrast, training networks identically but without constraints on symmetry yields richer dynamics. In particular, we show by analysing the network dynamics that the symmetric networks learn a stable code, whereas the unconstrained networks learn a code that is highly dynamic at the start of the delay period. This dynamic code is much closer to the electrophysiological data recorded from monkey PFC [45] during working memory tasks.

2 Methods

We trained Recurrent Neural Networks (RNNs) to perform a generic working memory task which requires the network, after a 'delay period', to correctly recall which one of the M initial conditions it started at. The networks receive no additional external input; their dynamics are determined entirely by the initial condition, and subsequent noise.

2.1 Recurrent Neural Network Dynamics

We study RNNs with $N = 50$ recurrent units and where each unit can emit both positive or negative connections (i.e. the units do not obey Dale's law). A tanh nonlinearity is used. The following vectorised differential equation governs the N dimensional neural activity $\mathbf{x}(t) = [\mathrm{x}_1(t), \mathrm{x}_2(t), \dots, \mathrm{x}_N(t)]$:

$$\tau \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = -\mathbf{x}(t) + \mathbf{W}_{\text{rec}}\mathbf{f}(\mathbf{x}(t)) + \sigma\boldsymbol{\eta}(t), \tag{1}$$

where $\mathbf{f}$ is a tanh nonlinearity acting on each element of $\mathbf{x}$, and τ is the time constant of the system. The network noise $\boldsymbol{\eta}(t)$ is identically and independently distributed (i.i.d.) over time, with each noise vector drawn from a zero-mean multivariate Gaussian with identity covariance matrix, and σ is a parameter that scales the noise. We use $N = 50$, $\tau = 50\,\text{ms}$, and $\sigma = \frac{0.01}{\sqrt{2}}$ throughout. $\mathbf{W}_{\text{rec}}$ is the $N \times N$ network connectivity matrix, and each element is initialised by sampling a zero-mean Gaussian with variance $1/N$, so that its largest eigenvalue is approximately one [50].

Euler integration is used to simulate the dynamics in equation 1. We are interested in how network dynamics vary for the same network given different initial conditions, so we shall denote network activity that was initialised at the m^{th} initial condition as $\mathbf{x}_m(t)$. Using the first-order approximation $\mathbf{x}(t+\delta t) \approx \mathbf{x}(t) + \delta t \left[\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}\right]$ we update the neural activity at each timestep according to

$$\mathbf{x}_m(t+\delta t) = \mathbf{x}_m(t) + \frac{\delta t}{\tau}\left[-\mathbf{x}_m(t) + \mathbf{W}_{\text{rec}}\mathbf{f}(\mathbf{x}_m(t)) + \sigma\boldsymbol{\eta}_m(t)\right] + \sqrt{\frac{\delta t}{\tau}}\left[\sigma\boldsymbol{\eta}_m(t)\right], \tag{2}$$

for each of the M initial conditions. $\delta t = 1\,\text{ms}$ throughout. Each initial condition is a network state $\mathbf{x}_m(t = 0)$ that corresponds to one of the values of θ_{cond}, which is the value of the stimulus variable, e.g. the angle of the cue in the memory guided saccade.

2.2 Task Setup

The working memory task we simulated is a classic memory guided saccade, although all our analysis is generic enough that it applies to any working memory task for a cyclic variable partitioned into M classes. The possible locations of the visual cue are constrained to lie evenly spaced on a circle, so that each stimulus position can be encoded by a single angle, which we denote θ_{cond}. We do not simulate a cue period, instead we use the initial condition to capture the cue specific activity, i.e. our simulation begins at the start of the delay period. The duration of the delay period is 1500 ms.

This is the same task used by Wimmer et al. [19] and others [11], [45] which uses $M = 8$ different cue directions. Other studies have used more classes, for example $M = 16$ [36]. This task may readily use entirely continuous initial conditions, as in Bays et al. [37] (with a colour wheel rather than a spatial task), in which case the objective is to minimise angular error rather than classification error.

Since the initial condition for the network captures the cue specific activity, the network can learn to solve the task simply by discretely classifying the different neural dynamics from each initial position. From the perspective of the network, this objective is equivalent to asking *'use the final state of this network to determine which one of the M states this network was initialised with'*. However, this becomes increasingly challenging as M increases and the initial conditions become more closely spaced. A more intuitive approach to the task is to store θ_{cond} as a continuous value in working memory for the duration of the delay. This approach does not become more difficult for increasing M. The cost function we shall present favours the latter representation of the initial condition.

2.3 Rings Embedded in High Dimensions

The possible positions of the stimulus form a circle, with each position representing a different value of θ_{cond}. The positions are evenly spaced, i.e. $\theta_{\text{cond}} = m\Delta\theta$ for $m \in \{1, 2, \ldots, M\}$ and $\Delta\theta = \frac{2\pi}{M}$. This circle is two dimensional, but the network state is N-dimensional, so to encode this initial angle in the network we first choose a two-dimensional subspace of $\mathbb{R}^N$, which is the plane spanned by the unit vectors $\hat{\mathbf{e}}_1^{\text{init}}$ and $\hat{\mathbf{e}}_2^{\text{init}}$. These vectors are randomly initialised, with the value for each dimension drawn from a zero-mean Gaussian, before each is then scaled to have unit norm. A ring of initial conditions of radius r is then created by rotating these vectors about the origin as follows;

$$\mathbf{x}_m(t = 0) = r \left[\sin(m\Delta\theta), \cos(m\Delta\theta)\right] \left[\hat{\mathbf{e}}_1^{\text{init}}, \hat{\mathbf{e}}_2^{\text{init}}\right]^\top, \tag{3}$$

for each $m \in \{1, 2, \ldots, M\}$. We empirically determine $r = 5$ to be a reasonable scaling for initial conditions.

2.4 Cost Function

We devised a cost function to train the parameters of the network to form a continuous attractor network. We introduce a $2 \times N$ (rank 2) matrix of 'readout weights', $\mathbf{W}_{\text{out}}$, and for the dynamics of each initial condition, take the inverse tangent of the two components resulting from this mapping to produce an angle, denoted $\theta_m(t)$. The two rows of $\mathbf{W}_{\text{out}}$ are written $\mathbf{w}_1$ and $\mathbf{w}_2$, and each is initialised by drawing values from a standard Gaussian. The cost function is then computed over time and initial conditions using the cosine of the difference in angle between $\theta_m(t)$, and the angle that was encoded by the initial condition of these dynamics, $m\Delta\theta$. The cost function is

$$\mathcal{C}(t) = 1 - \frac{1}{M}\sum_{m=1}^{m=M} \cos(\theta_m(t) - m\Delta\theta), \tag{4}$$

where

$$\theta_m(t) = \measuredangle\left([\mathbf{w}_1, \mathbf{w}_2]^\top \mathbf{x}_m(t)\right), \tag{5}$$

and $\measuredangle(\boldsymbol{\alpha})$ gives the angle between $\boldsymbol{\alpha}$ and the vector $[1, 0]$.

This cost function is applied to the final 1000 ms of the delay period, $T_1 = 500$ ms to $T_2 = 1500$ ms, so the total cost of the network dynamics is

$$\mathcal{C}_{\text{tot}} = \int_{T_1}^{T_2} \mathcal{C}(t)dt. \tag{6}$$

2.5 Training Attractor Networks

The parameters of the network that are allowed to train to minimise the cost are the network weights, readout weights, and the unit vectors defining the initial ring; i.e., $\mathbf{W}_{\text{rec}}, \mathbf{W}_{\text{out}}, \hat{\mathbf{e}}_1^{\text{init}}, \hat{\mathbf{e}}_2^{\text{init}}$. Importantly, we constrain some networks to have symmetric $\mathbf{W}_{\text{rec}}$, by constructing $\mathbf{W}_{\text{rec}} = \frac{1}{2}\left(\mathbf{W}_{\text{sym}} + \mathbf{W}_{\text{sym}}^\top\right)$ and only directly training $\mathbf{W}_{\text{sym}}$. We train these symmetric networks as a control to compare the effect of constraining network structure in this fashion, as most existing networks are symmetric [19], [35], [39].

To train each RNN, the network is first set to its initial conditions, then dynamics are run, and the cost function evaluated at each timestep. The values of the cost at each timestep are summed to generate the overall cost, approximating the integral in equation 6. The network and cost function are implemented in Tensorflow in Python, and automatic differentiation is used with an Adam optimiser (with a learning rate of 0.0005) to minimise the cost with respect to the set of trainable network parameters. A batch size of 25 is used throughout; each batch has independently generated noise and the cost function is averaged over batches. Each computation of the network cost in equation 6 and subsequent gradient update is one epoch. Each network was trained for 1,000 epochs, with a criteria that if the cost exceeds 0.5 (i.e. a mean angular error of $\pi/3$) at epoch 100 then the network is discarded and training restarted with all randomly initialised parameters redrawn. This happens in around 10% of training attempts. Successfully trained networks typically have a cost between 0.001 and 0.005.

2.6 State-Space Trajectories

Whilst it is possible to directly plot the activities of all neurons in these networks, this is not usually the most insightful way to visualise neural dynamics. Instead, we use principal component analysis (PCA), which is widely used for dimensionality reduction of neural dynamics [51]–[53]. The dynamics of a network are a path in N-dimensional state-space, which we visualise by projecting onto a two-dimensional subspace. The subspace that we project onto is that spanned by the top two principal components (PCs) of the neural activity during the final 500 ms ('late-delay'). We call this plane the 'attractor space', and denote the first and second principal components as $\mathrm{PC}_1^{\mathrm{attr}}$ and $\mathrm{PC}_2^{\mathrm{attr}}$ respectively. These PCs form axes in figures 5, 6, 7, 8, 9, 10, 11.

2.7 Network Slowness

There is meaningful analysis we can undertake that focuses on the network itself, rather than simulated dynamics. A simple way to directly analyse attractor properties of a network, is to plot the 'slowness' of trajectories in state space. In equation 1, $\mathbb{E}[\boldsymbol{\eta}(t)] = \mathbf{0}$, so $\mathbb{E}[\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}] \propto -\mathbf{x}(t) + \mathbf{W}_{\mathrm{rec}}\mathbf{f}(\mathbf{x}(t))$. Accordingly, for any point in the state-space of the network $\mathbf{x}$ we define the slowness to be the two-norm $\|\mathbf{g}(\mathbf{x})\|_2$, where $\mathbf{g}(\mathbf{x}) = -\mathbf{x} + \mathbf{W}_{\mathrm{rec}}\mathbf{f}(\mathbf{x})$. To visualise the slowness we sample points from an evenly-spaced grid on the attractor space and evaluate $\|\mathbf{g}(\mathbf{x})\|_2$ at every point, giving a 'heatmap' of network slowness.

2.8 Fixed Point Analysis

To determine the locations of the attractor states of a network a numerical method is used that attempts to solve for the solutions of $\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{0}$. As the noise for this system has zero mean, it does not affect the location of any attractor states of a given network, and so is removed by setting the noise scaling to $\sigma = 0$. From eq. 1, $\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{0}$ then requires $\mathbf{x} = \mathbf{W}_{\mathrm{rec}}\mathbf{f}(\mathbf{x})$, i.e. the fixed points of $\mathbf{W}_{\mathrm{rec}}\mathbf{f}(\mathbf{x})$ are to be found. Note that from the evenness of $\mathbf{f} = \tanh$, $\mathbf{g}$ is an even function of $\mathbf{x}$, and so a fixed point at $\mathbf{x}$ implies a fixed point at $-\mathbf{x}$. As in section 2.7, $\mathbf{g}(\mathbf{x})$ is the expression $-\mathbf{x} + \mathbf{W}_{\mathrm{rec}}\mathbf{f}(\mathbf{x})$.

A set of $P = 1000$ 'particles' are independently sampled from a distribution over the state-space of the system, which is an N-dimensional multivariate Gaussian with mean zero and identity covariance matrix. Each particle is scaled by a constant so that the 'cloud' of particles is sufficiently spread out so as to avoid too many particles being too close to the trivial attractor state $\mathbf{x} = \mathbf{0}$. This scaling constant was empirically set to 10 for all analyses. A cost function is then implemented, and minimised with respect to the particles $\mathbf{x}_p$;

$$\mathcal{C} = \sum_{p=1}^{p=P} \|\mathbf{g}(\mathbf{x}_p)\|_2^2. \tag{7}$$

Iterative minimisation of this cost causes the particles to travel through state space towards locations that have slower dynamics. This optimisation is run for 5000 iterations, with the learning rate of the Adam optimiser, which starts at 0.1, halved every 1000 iterations as the speed near the fixed points asymptotically approaches zero. Particles which have speed above a suitably low threshold (0.005) are discarded, and all remaining points are then determined to be sufficiently slow that they are effectively attractor states. The trajectories that each particle takes during this minimisation can also provide insight to the behaviour of the network.

2.9 Cross-Temporal Decodability

We can determine how well a network solves our task by training a classifier to discriminate between dynamics that started at different initial conditions. To understand the nature of the coding used by the network, we analyse this classification performance in a fully cross-temporal fashion. Previous research has used this same decoding analysis [44], [46] (section 1.3, figure 4).

For a given model we generate 10 batches of neural activity, each initialised identically but with independently generated noise. We assign 5 batches for training and 5 for testing. We partition neural activity into non-overlapping 25 ms bins. We then arbitrarily assign class labels so that each sample of N-dimensional activity has a label corresponding to its initial condition M. We train logistic regression classifiers in Python using `sklearn.linear_model.LogisticRegression`, with the maximum number of iterations set to 5,000 and all other parameters as default. To generate element ij of the full cross-temporal decodability matrix, we train a classifier on the i^{th} bin of the training batches, and test on the j^{th} bin of the testing batches. The 'decoding accuracy' is the fraction of activity samples that classified to the correct value of M.

2.10 Alignment Index

We measure the 'alignment index', set out in [54], to measure how the PCs of neural activity at different times align with each other. This is performed in a fully cross-temporal fashion. We start by assigning 5 training and 5 testing batches, and partition dynamics into 25 ms bins exactly as in section 2.9. We then take the top two PCs of the training bin at t_1, denoted $\text{PC}_1^{(t_1)}$ and $\text{PC}_2^{(t_1)}$, and compute what fraction of the variance of the testing bin at t_2 is explained by these components. If $\boldsymbol{P} = [\text{PC}_1^{(t_1)}, \text{PC}_2^{(t_1)}]$ and $\boldsymbol{C}(t)$ is the covariance (across batches, initial conditions, and timesteps) of the testing bin at time t then we can write the alignment index matrix as

$$\boldsymbol{A}_{t_1 t_2} = \frac{\text{Tr}(\boldsymbol{P}(t_1)^T \boldsymbol{C}(t_2) \boldsymbol{P}(t_1))}{\text{Tr}(\boldsymbol{C}(t_2))}, \tag{8}$$

where Tr is the matrix trace operator.

3 Results

RNNs were trained according to the cost function detailed in equation 6. After confirming that the networks behave as expected for low values of M, the value of M was increased until it was determined that the nature of the attractor states of the networks produced were continuous rather than discrete. For $M = 20$, we then trained ten symmetric RNNs and ten unconstrained RNNs as described in the methods and analysed their properties.

3.1 Trained Networks Solve the Task

Our first results are to verify that training networks results in the task being solved. We set $M = 2$, which corresponds to a simple left vs right memory guided saccade. A single unconstrained network was randomly initialised, and trained for 50 epochs. The attractor space PCs were taken for the trained network, and the activity of both the untrained and trained network plotted in this space. Figure 5 shows 1500 ms of neural dynamics for this network, starting from each initial condition.

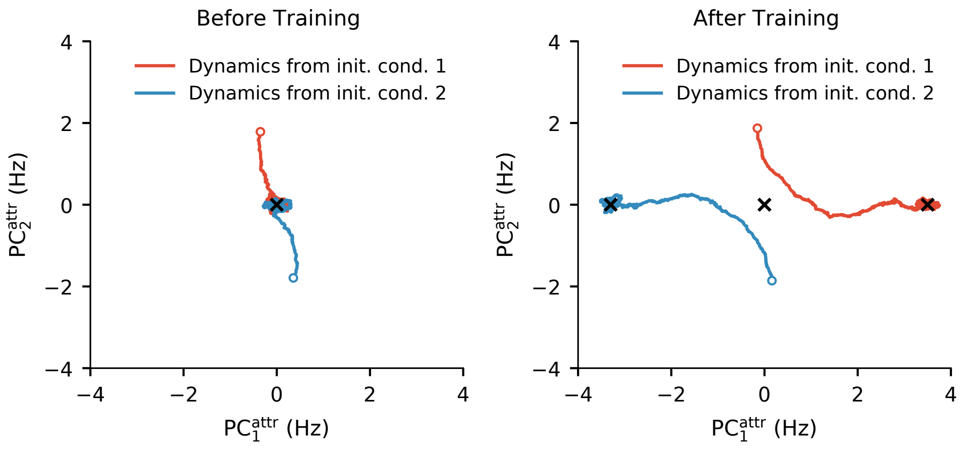

Figure 5: 1500 ms of activity projected into the attractor space for an unconstrained network with $M = 2$. Open circles correspond to $t = 0$. Before training dynamics have poor decodability, particularly at the end of the delay; after training, dynamics from each of the initial conditions are decodable, and two attractor states have been formed.

Before training, the dynamics show both initial conditions decaying to zero, after which they are effectively indistinguishable from each other. Conversely, the state-space trajectories taken from each initial condition for the trained network are easily separable, as shown in figure 5. Therefore the trained network displays easily decodable dynamics and has solved the task.

3.2 Trained Networks Display Attractor States

The solution of the task is insufficient to conclude that an attractor network has been formed. To confirm that our trained networks do display attractor states, we trained an unconstrained network with $M = 8$ and undertook the fixed point analysis detailed in section 2.8.

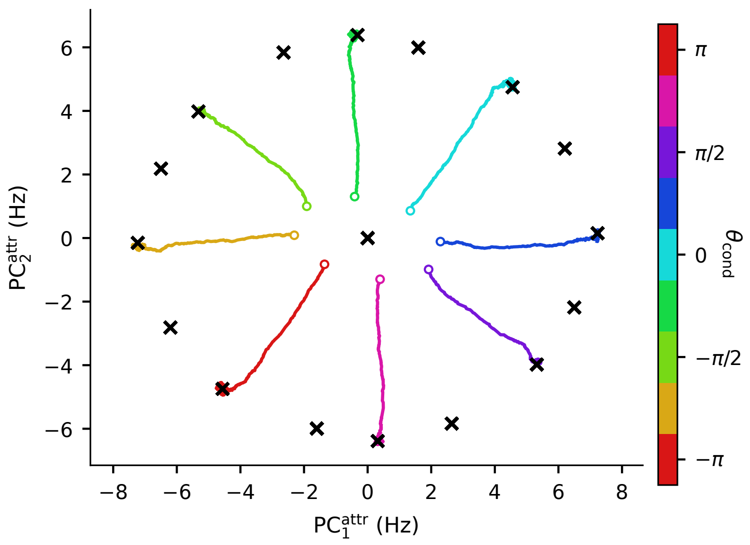

Figure 6: Neural dynamics of a network trained with $M = 8$. Open circles correspond to $t = 0$. Attractor states, marked with crosses, are created for each initial condition, with these points lying on a ring. Crosses mark the determined location of attractor states, from fixed point analysis.

The attractor states formed by training this network are shown in figure 6, overlayed onto the neural dynamics for each initial value of θ_{cond} (plotted in the same space spanned by the top two PCs of late neural activity). The points at which the neural dynamics are drawn towards, and stagnate at, exactly match the determined locations of attractor states. The attractor states lie on a ring that is embedded in the N-dimensional network state. The evenness of $\mathbf{g}$ (see section 2.7) means each attractor state has a conjugate attractor state π radians around the ring.

A crucial question to ask of the network in figure 6 is whether it forms a continuous or discrete attractor network. The attractor states marked in figure 6 do lie on a ring, but despite the fixed point analysis using $P = 1000$ particles (684 of which pass below the slowness threshold, see section 2.8) only 16 individual attractor states are visible (excluding $\mathbf{0}$). Each cross is in fact dozens of particles that have converged to the same point - so the attractor states are discrete.

This analysis demonstrates that our cost function produces networks with attractor states, despite the cost function scoring only a general requirement of the task, and making no assumptions about whether or not attractor states should form.

3.3 From Discrete to Continuous Attractor Networks

We have shown we can train attractor networks with discrete attractor states, but can we train attractor networks with continuous attractor manifolds?

We investigate further the nature of the attractor states of the previous network (figure 6) by plotting the trajectories of all particles during fixed point analysis, and also the slowness of the network (figure 7). The increased prevalence of particle trajectories lying on the ring, between the discrete attractor states (the crosses in figure 6), indicates that during fixed point analysis the particles are first drawn onto the ring, then slowly move along the ring towards local minima. The ring in the slowness plot of figure 7 is uneven, further showing that the speed is inconsistent at different positions on the ring. This reaffirms our conclusion that this network has formed a discrete attractor.

Despite this, both plots clearly show that points lying on the ring tend to be slower than points not on the ring. It is remarkable that with as few as $M = 8$ initial conditions, discrete attractor states already begin to merge to form a continuum. Current working memory models require convoluted recipes [19], [30] to produce networks capable of solving this task, even for $M = 8$ [19]. This demonstrates the generality of our cost function and the power of using a normative approach over developing such models by hand; we make fewer assumptions about the structure of the network weights (section 1.2), and are able to produce more realistic networks as a result.

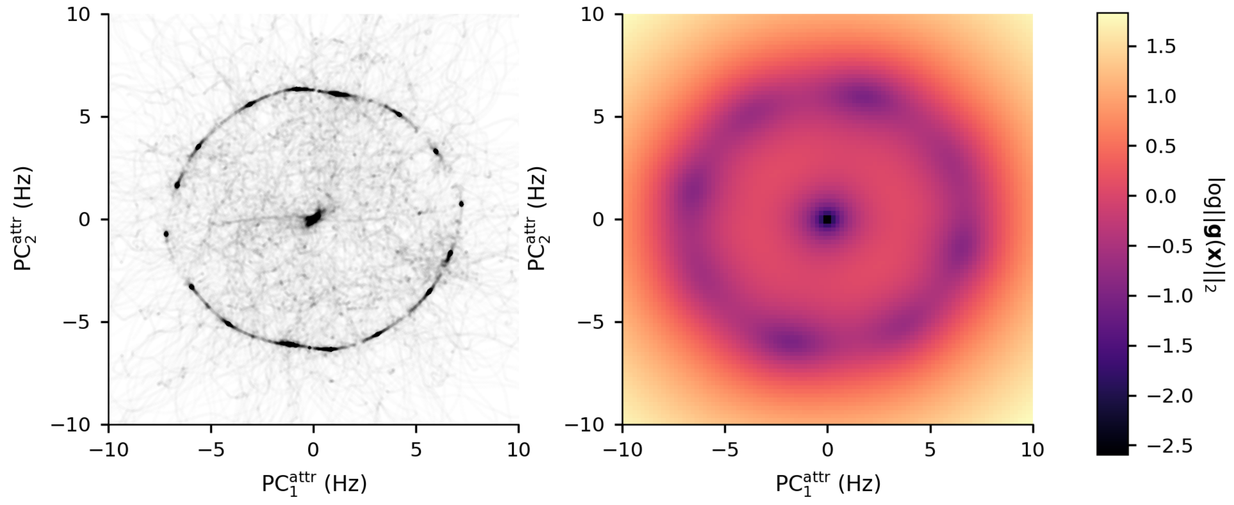

Figure 7: Analysis of unconstrained network trained with $M = 8$. Left: fixed point analysis particle trajectories plotted with transparency. The dark patches on the ring each correspond to the endpoints of many particles, i.e. the crosses in figure 6. Right: log plot of the network speed $\|\mathbf{g}(\mathbf{x})\|_2$. As in section 2.8, $\mathbf{g}(\mathbf{x}) = -\mathbf{x} + \mathbf{W}_{\text{rec}}\mathbf{f}(\mathbf{x})$.

We now demonstrate that by increasing the number of initial conditions M, the many discrete attractor states spaced on the ring merge to form one continuous ring. We trained networks with values of M ranging from 4 to 20 and plotted the slowness of each network in its attractor space (figure 8).

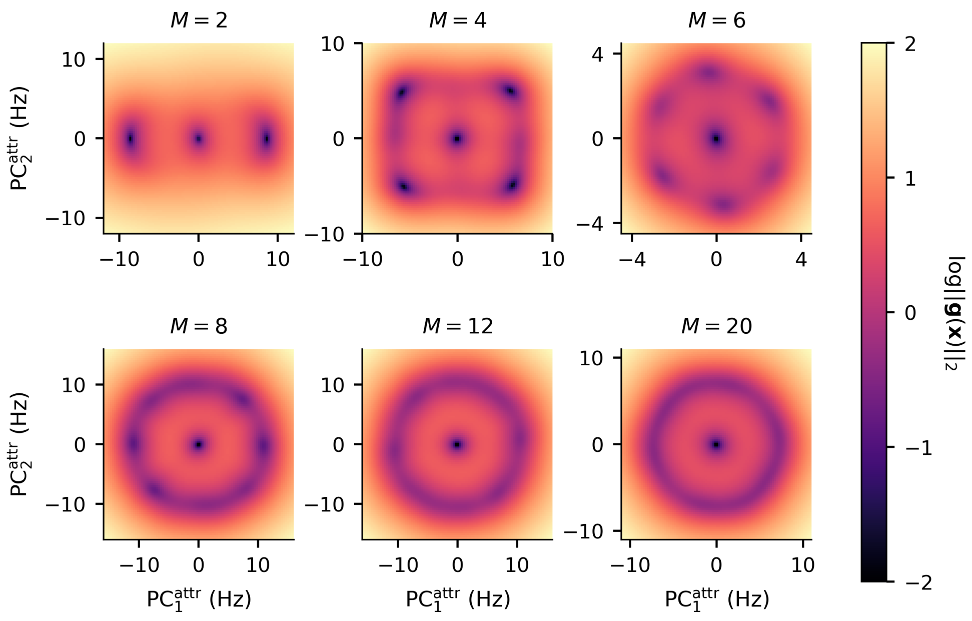

Figure 8: Slowness of six different unconstrained networks with varying numbers of initial conditions M. Each plot is normalised onto the same scale. As M increases, the nature of the network transitions from a discrete attractor to a continuous attractor.

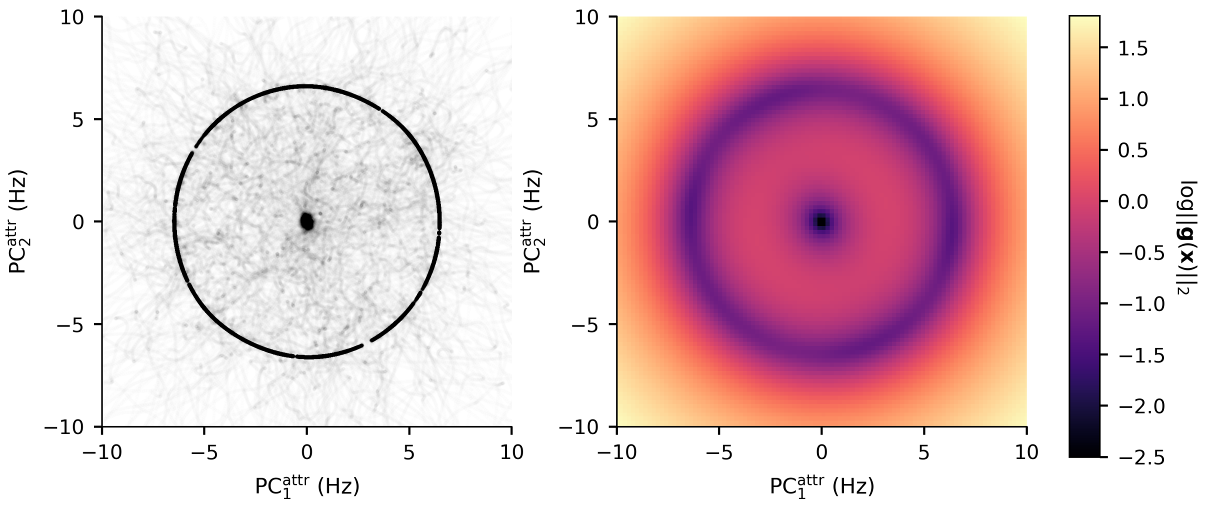

Figure 9: Unconstrained network trained with $M = 20$ (not the same network as in figure 8). Left: fixed point analysis particle trajectories, with the same number of particles, size, and transparency, as figure 7. Right: log plot of the network speed $\|\mathbf{g}(\mathbf{x})\|_2$. As before, $\mathbf{g}(\mathbf{x}) = -\mathbf{x} + \mathbf{W}_{\text{rec}}\mathbf{f}(\mathbf{x})$.

As discussed in the previous section, the formation of a ring begins remarkably quickly with respect to M, even for $M = 4$ there is some clear continuity, and not simply four unrelated attractor states. We suggest that this results from the symmetry of the initial conditions, i.e. their positioning on a ring, though no tests are performed on non-ring initial conditions to confirm this with certainty.

We find that by $M = 20$, the slowness plot shows a smooth ring, and we undertake fixed point analysis of a $M = 20$ unconstrained network to determine if the attractor states are genuinely continuous. We plot the trajectories of particles during fixed point analysis, and the slowness of the network in the exact same fashion as in figure 7. In stark contrast to the earlier $M = 8$ network (figure 6), the $M = 20$ network displays a clear continuum of fixed points; the particle trajectories of fixed point analysis do not bunch together into discrete points and are evenly distributed around the ring (figure 9). A further plot of the speed-thresholded end locations of the fixed point analysis particles on another two unconstrained networks trained in the same way confirms that these trajectories are not drawn towards a small number of discrete states but are spread almost perfectly uniformly across the ring (figure 10). These are continuous attractor networks.

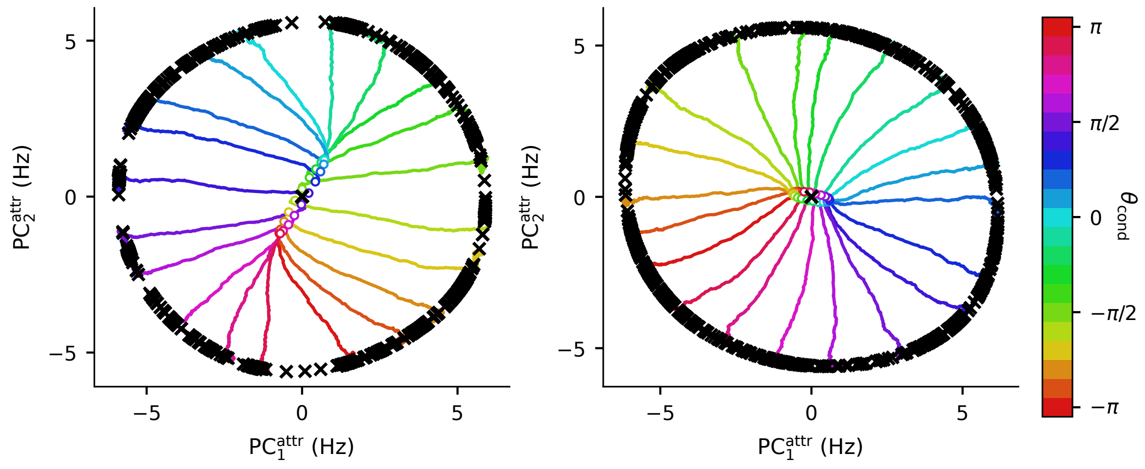

Figure 10: Neural activity of two example unconstrained networks trained with $M = 20$. Open circles mark $t = 0$ and attractor states are marked with crosses. Attractor states are found in a continuum around the ring rather than just at the end points of the dynamics. There are clearly significant dynamics in directions orthogonal to the attractor space.

Figures 9 and 10 show that our cost function, as well as producing discrete attractor networks for low M, can produce continuous attractor networks by increasing M to 20. As before, our cost function requires no more than the maintenance of a variable during the delay period - the emergence of continuous attractors from random networks as a result of *solving a working memory task* rather than by deliberate and unrealistic manipulation of the network weights $\mathbf{W}_{\text{rec}}$ provides a more plausible explanation for how such a circuit might actually arise in the brain.

3.4 Dynamics of Symmetric and Unconstrained Networks

We now compare the neural dynamics of unconstrained networks with $M = 20$ to networks that are trained otherwise identically but with the constraint that $\mathbf{W}_{\text{rec}}$ must be symmetric (figure 11).

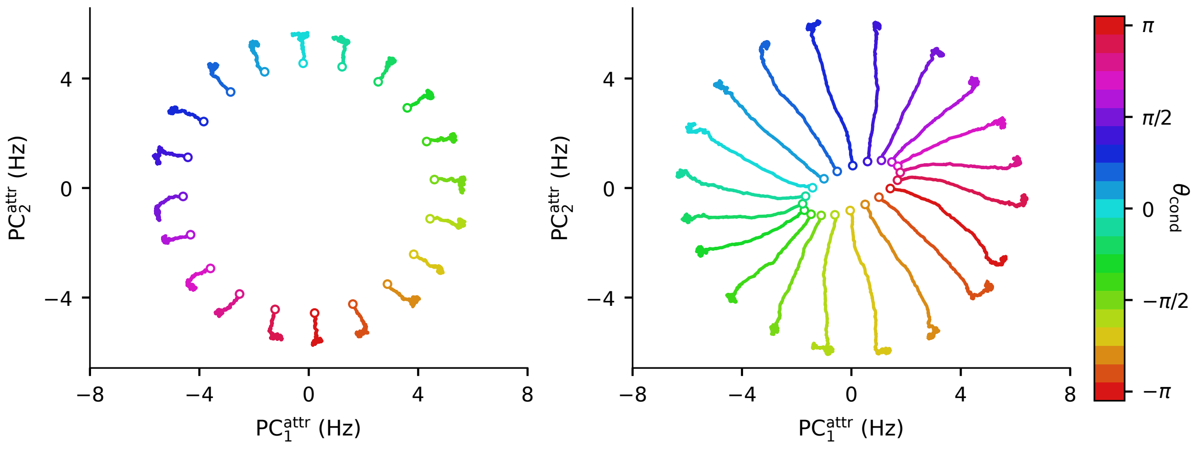

Figure 11: Dynamics of models trained with $M = 20$ initial conditions. Open circles correspond to $t = 0$. Left - symmetric network, right - unconstrained network. The symmetric network learns to solve the task by staying very close to its initial conditions, precluding a strongly dynamic code. The unconstrained network displays more interesting dynamics, which may be due to dynamic coding.

The key result of figure 11 is that the symmetric networks learn to solve the task in using the same approach that existing models, which mostly are symmetric [19], do. That is, as discussed in section 1.3, the task is solved effectively by remaining at the location of the initial conditions for the duration of the delay period. If the network remains in one state for the duration of the delay period, it must be using stable coding.

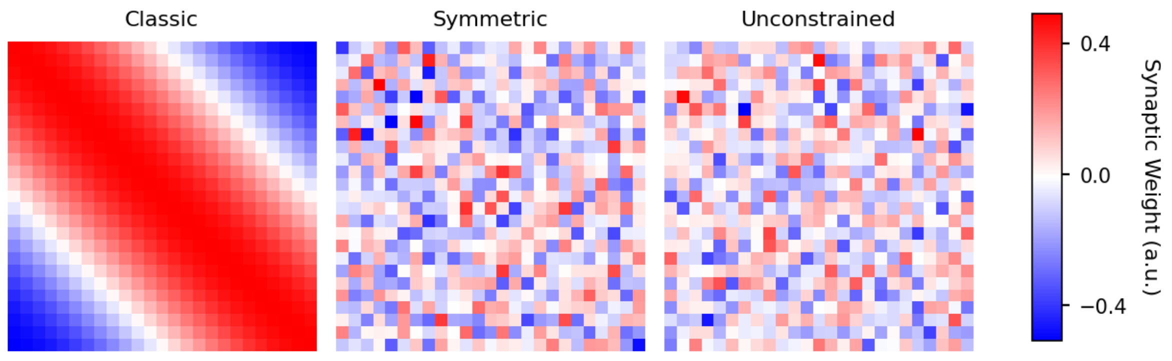

Figure 12: Network connectivity for the first 25 of $N = 50$ recurrent units, i.e. top-left quadrant of $\mathbf{W}_{\text{rec}}$, for three networks. Left to right: classic-style model using cosine function, trained symmetric network, trained unconstrained network.

Nevertheless, the creation of symmetric continuous attractor networks from a random initial condition is an important result. Firstly, training networks in this fashion does not result in a synaptic connectivity matrix (figure 12) with the rigid structure that we observe in classic models [17], [19], [30]. (The differences in structure are not explained by the permutation of rows and columns, which is allowed because the ordering of neurons is arbitrary.) Secondly, rather than being constructed from a pre-planned neural circuit design, these networks can be generated by optimisation of the cost function which only requires the working memory task to be solved, and does not place any requirements on the network structure that should emerge - providing a more plausible explanation for how such circuits might actually arise in the brain.

3.5 Evidence for Dynamic Coding

Figure 11 appears to show that when we train symmetric models, the result is a stable coding. To determine robustly the nature of the coding in the unconstrained case we investigate the dynamics of these networks further. The first and most simple analysis we undertake is to directly plot neuron activity (figure 13) for both of the networks in figure 11. The unconstrained network demonstrates a strongly dynamic code; the state at the start of the delay period is very different to the state at the end (recall the toy example showing precisely this in figure 3).

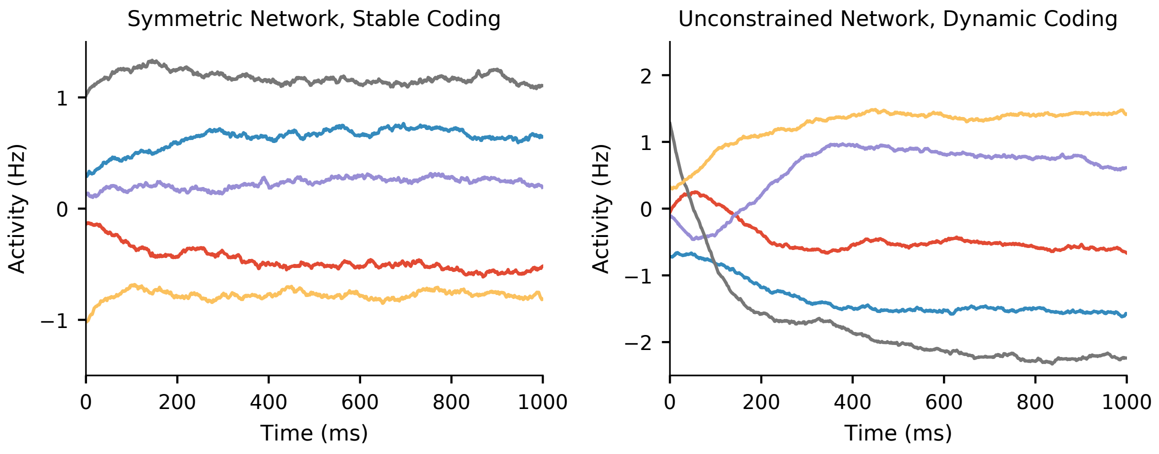

Figure 13: Examples of the first 1000 ms of neuron activities of symmetric and unconstrained networks, for a single initial condition. The first five neurons are shown as a representative sample of the network. The symmetric networks exhibit stable coding, whereas the unconstrained network appears to exhibit dynamic coding.

A more thorough analysis of the decodability of unconstrained networks was then performed. We trained classifiers on late delay dynamics to measure how well the code used by the network at the end of the delay generalises to the dynamics at the start. This is exactly the top row of the full cross-temporal decodability matrix (section 2.9, figures 4, 15). For $M = 20$, we trained 10 symmetric and 10 unconstrained networks and averaged decoder accuracy (figure 14). The decrease in decoding accuracy by the unconstrained networks at early-delay times clearly shows that by not requiring our networks to be symmetric, the solution to our cost function that the networks find is one using dynamic coding.

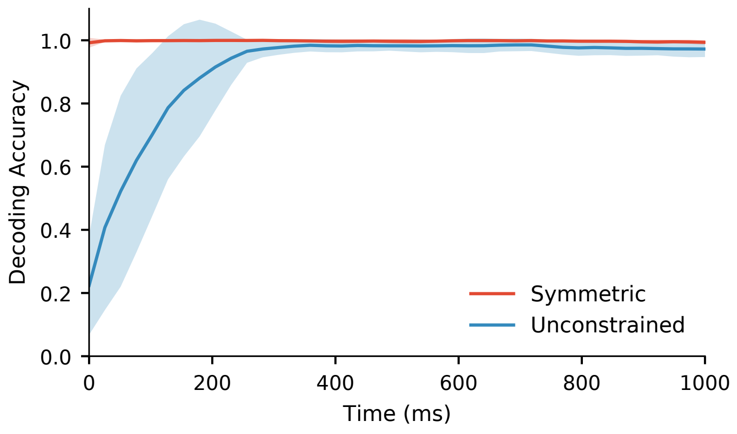

Figure 14: First 1000 ms of decodability of dynamics using classifiers trained on late-delay dynamics (trained separately for each model), for 10 unconstrained and 10 symmetric networks, for $M = 20$. The mean is taken across batches and networks, with the shading showing $\pm$ one standard deviation. Note that the standard deviation for symmetric networks is barely visible as decoding accuracy is close to unity for all networks.

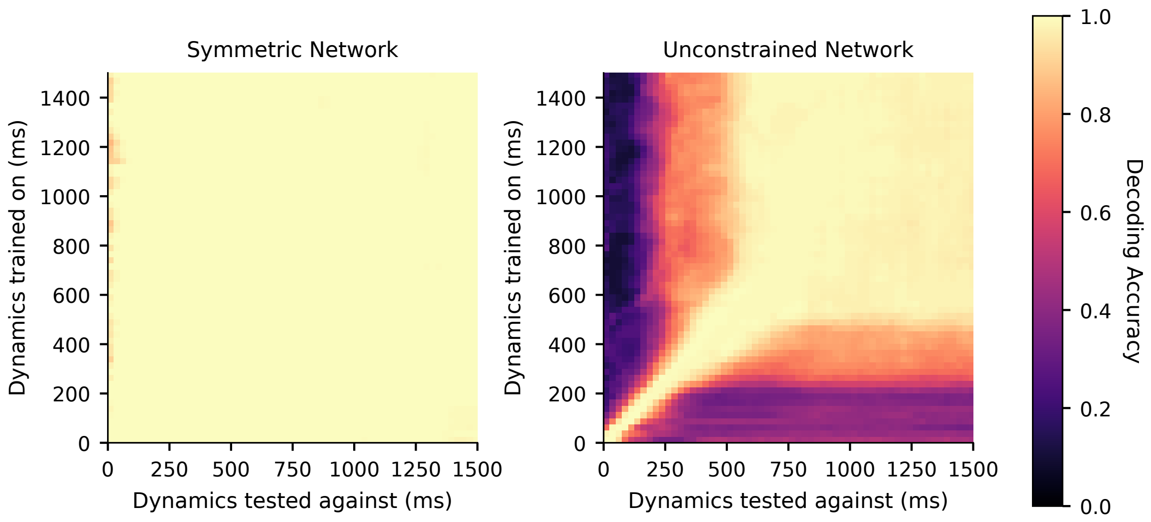

Figure 15: Full cross-temporal decodability for a single symmetric network (left) and unconstrained network (right). The unconstrained network displays dynamic coding, with a solid block of late-delay decodability, i.e. the same code from that time onwards.

To allow a comparison of our simulations to the electrophysiological data from the monkey PFC, presented by Spaak et al. [45] and others [46], [47], we plot the full cross-temporal decodability matrix for a single unconstrained network (figure 15). As a control, we also plot the cross-temporal decodability for a single symmetric network, to highlight the inability of existing models to explain dynamic coding, due to their symmetry. The symmetric network shows a block of near-perfect decodability for all pairs of times - this is what we expect from a network that uses a stable code, and solves the task by effectively remaining at the initial condition for the duration of the delay period. The unconstrained network produces a cross-temporal decodability matrix that is phenomenologically the same as the recordings from monkeys in figure 4; it displays a strongly dynamic coding, but good generalisation of the late-delay code.

Our final analysis is to plot the alignment index, which is the cross-temporal measures of the fraction of the variance at time t_2 explained by the top two PCs at t_1 (section 2.10). The alignment index is plotted in figure 16, for the same pair of networks that are shown in figure 15. The alignment index is a meaningful metric because it quantifies how the space in which the attractor network is encoding information, varies over time. We initialise these networks with initial conditions lying on a ring (section 2.3) and observe that the attractor space formed is a ring (figure 11) - but the planes of these rings are not necessarily aligned.

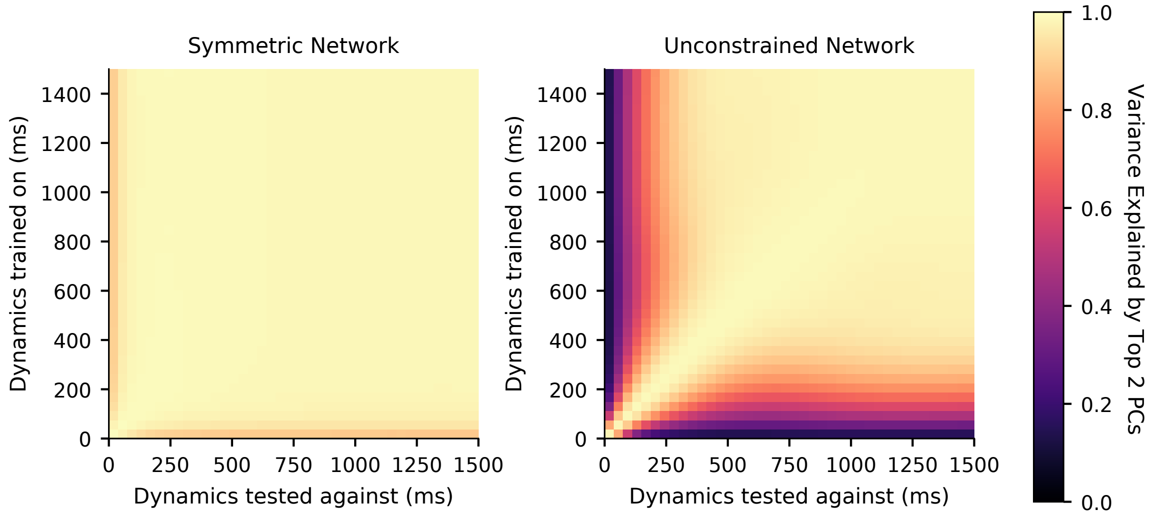

Figure 16: Alignment index for one symmetric (left) and one unconstrained (right) network, the same networks as figure 15

Figure 16 demonstrates that in the symmetric network case, the plane that the ring of initial conditions lies in (i.e. $\hat{\mathbf{e}}_1^{\text{init}}, \hat{\mathbf{e}}_2^{\text{init}}$) are well aligned with the plane of the attractor space (i.e. $\text{PC}_1^{\text{attr}}, \text{PC}_2^{\text{attr}}$), because the same principal components can capture nearly all the variance for all time. For the unconstrained network, the activity remains primarily in 2 dimensions for the entire delay period (the diagonal is close to one), but the plane of the initial ring is orthogonal to the attractor space. During the delay period, the plane that the network encodes θ_{cond} in rotates (in M-dimensional space) to align with the attractor space until its dynamics are well explained by the two PCs of the final dynamics, i.e. the block of alignment index near one after the initial transient in figure 16.

These results represent very strong evidence of dynamic coding in our unconstrained attractor neural networks. The dynamic coding is directly visible in the activity of the network (figure 13), and cross-temporal decodability (figures 14, 15) reaffirms this by showing that the neural codes used by our networks consistently generalise poorly from early to late-delay periods and vice versa. The dynamic aspect of this coding in our model can be understood as the attractor network encoding a continuous variable θ_{cond} as a location on a 2-dimensional ring for the duration of the delay period, where importantly the plane of the ring rotates during network dynamics from its initial orientation, until the orientation of ring is such that it forms a continuous ring of attractor states.

4 Conclusions

- Hand-crafted rigid structure in network connectivity matrices is not necessary to produce continuous attractor networks, which can be learned from random initial conditions.
- Normative methods provide a powerful means of creating networks with particular dynamics, that would otherwise be difficult or impossible to hand-tune.
- Continuous attractor networks can form even if the network is only required to store a discrete set of inputs, provided that sufficiently many inputs from a continuous domain are used.
- Continuous attractor network models constrained to have symmetric connectivity matrices display stable coding when trained on our working memory maintenance task, and solve the task by staying in effectively the same state for the duration of the delay period
- Continuous attractor network models with no constraints on connectivity matrices can learn to solve our working memory maintenance task by encoding information as a location on a ring; the basis of this ring can dynamically vary throughout our delay period, facilitating dynamic coding.

5 Future Work

- Extending our networks to use more realistic nonlinearities such as ReLU or rectified parabola rather than tanh, which unrealistically saturates (but we found easier to train).
- Extending our single $\mathbf{W}_{\text{rec}}$ matrix to fully EI networks.
- Investigating the tuning curves of neurons in our networks to determine if unconstrained networks produce heterogeneous tuning curves.
- More sophisticated mathematical analysis of the attractor manifold, such as showing (by linearising our dynamics equations) that speed orthogonal to the manifold is higher than speed along it.

References

[1] R. Chaudhuri and I. Fiete, "Computational principles of memory," *Nature Neuroscience*, vol. 19, pp. 394–403, 2016.

[2] N. Cowan, "What are the differences between long-term, short-term, and working memory?" *Progress in brain research*, vol. 169, pp. 323–338, 2008.

[3] N. Cowan, "Working memory underpins cognitive development, learning, and education," *Educational Psychology Review*, vol. 26, pp. 197–223, 2014.

[4] S. Funahashi, "Working memory in the prefrontal cortex," *Brain Sciences*, vol. 7, p. 49, 2017.

[5] C. Bailey, E. Kandel, and K. Harris, "Structural components of synaptic plasticity and memory consolidation," *Cold Spring Harbor perspectives in biology*, vol. 7, a021758, 2015.

[6] W. B. Scoville and B. Milner, "Loss of recent memory after bilateral hippocampal lesions," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 20, no. 1, pp. 11–21, 1957.

[7] C. Tetzlaff, C. Kolodziejski, I. Markelic, and F. Wörgötter, "Time scales of memory, learning, and plasticity," *Biological cybernetics*, vol. 106, pp. 715–726, 2012.

[8] C. Koch, M. Rapp, and I. Segev, "A brief history of time (constants)," *Cerebral cortex (New York, N.Y. : 1991)*, vol. 6, pp. 93–101, 1996.

[9] A. Destexhe, Z. Mainen, and T. Sejnowski, *Kinetic Models of Synaptic Transmission*. 1998, vol. 2, pp. 1–26.

[10] H. Inagaki, L. Fontolan, S. Romani, and K. Svoboda, "Discrete attractor dynamics underlies persistent activity in the frontal cortex," *Nature*, vol. 566, pp. 212–217, 2019.

[11] S. Funahashi, C. Bruce, and P. Goldman-Rakic, "Mnemonic coding of visual space in the monkeys dorsolateral prefrontal cortex," *J. Neurophysiol.*, vol. 61, pp. 1–19, 1989.

[12] P. Goldman-Rakic, "Cellular basis of working memory," *Neuron*, vol. 14, pp. 477–485, 1995.

[13] M. Kritzer and P. Goldman-Rakic, "Intrinsic circuit organization of the major layers and sublayers of the dorsolateral prefrontal cortex in monkeys," *The Journal of comparative neurology*, vol. 359, pp. 131–143, 1995.

[14] J. B. Levitt, D. A. Lewis, T. Yoshioka, and J. S. Lund, "Topography of pyramidal neuron intrinsic connections in macaque monkey prefrontal cortex (areas 9 and 46)," *Journal of Comparative Neurology*, vol. 338, no. 3, pp. 360–376, 1993.

[15] R. Lorente, "Vestibulo-ocular reflex arc," *Archives of Neurology & Psychiatry*, vol. 30, no. 2, pp. 245–291, 1933.

[16] J. Fuster and G. Alexander, "Neuron activity related to short-term memory," *Science (New York, N.Y.)*, vol. 173, pp. 652–654, 1971.

[17] X.-J. Wang, "Synaptic reverberation underlying mnemonic persistent activity," *Trends in Neurosciences*, vol. 24, no. 8, pp. 455–463, 2001.

[18] C. Ranganath and M. D'Esposito, "Directing the mind's eye: Prefrontal, inferior and medial temporal mechanisms for visual working memory," *Current opinion in neurobiology*, vol. 15, pp. 175–182, 2005.

[19] K. Wimmer, D. Nykamp, C. Constantinidis, and A. Compte, "Bump attractor dynamics in prefrontal cortex explains behavioral precision in spatial working memory," *Nature neuroscience*, vol. 17, pp. 431–439, 2014.

[20] G. Yang, M. Joglekar, H. Song, W. Newsome, and X.-J. Wang, "Task representations in neural networks trained to perform many cognitive tasks," *Nature Neuroscience*, vol. 22, 2019.

[21] A. Redish, A. Elga, and D. Touretzky, "A coupled attractor model of the rodent head direction system," *Network: Computation in Neural Systems*, vol. 7, pp. 671–685, 1996.

[22] S. Kim, H. Rouault, S. Druckmann, and V. Jayaraman, "Ring attractor dynamics in the drosophila central brain," *Science (New York, N.Y.)*, vol. 356, 2017.

[23] H. Seung, "Continuous attractors and oculomotor control. neural networks," *Neural networks : the official journal of the International Neural Network Society*, vol. 11, pp. 1253–1258, 1998.

[24] B. Mcnaughton, F. Battaglia, O. Jensen, E. Moser, and M.-B. Moser, "Path integration and the neural basis of the 'cognitive map.'," *Nature reviews. Neuroscience*, vol. 7, pp. 663–678, 2006.

[25] Y. Burak and I. Fiete, "Accurate path integration in continuous attractor network models of grid cells," *PLoS computational biology*, vol. 5, pp. 245–291, 2009.

[26] E. Rolls, "An attractor network in the hippocampus: Theory and neurophysiology," *Learning & memory (Cold Spring Harbor, N.Y.)*, vol. 14, pp. 714–731, 2007.

[27] T. Wills, C. Lever, F. Cacucci, N. Burgess, and J. O'Keefe, "Attractor dynamics in the hippocampal representation of the local environment," *Science (New York, N.Y.)*, vol. 308, pp. 873–876, 2005.

[28] C. Brody, R. Romo, and A. Kepecs, "Basic mechanisms for graded persistent activity: Discrete attractors, continuous attractors, and dynamic representations," *Current opinion in neurobiology*, vol. 13, pp. 204–211, 2003.

[29] K. Zhang, "Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory," *Journal of Neuroscience*, vol. 16, no. 6, pp. 2112–2126, 1996.

[30] R. Ben-Yishai, R. Bar-Or, and H. Sompolinsky, "Theory of orientation tuning in visual cortex," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 92, pp. 3844–3848, 1995.

[31] C. Constantinidis and X.-J. Wang, "A neural circuit basis for spatial working memory," *The Neuroscientist : a review journal bringing neurobiology, neurology and psychiatry*, vol. 10, pp. 553–565, 2005.

[32] P. Miller, "Analysis of spike statistics in neuronal systems with continuous attractors or multiple, discrete attractor states," *Neural computation*, vol. 18, pp. 1268–1317, 2006.

[33] D. Amit and N. Brunel, “Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex.,” vol. 7, pp. 237–252, 1997.

[34] V. Mante, D. Sussillo, K. Shenoy, and W. Newsome, “Context-dependent computation by recurrent dynamics in prefrontal cortex,” *Nature*, vol. 503, pp. 78–84, 2013.

[35] S. Wu, K. Hamaguchi, and S.-i. Amari, “Dynamics and computation of continuous attractors,” *Neural computation*, vol. 20, pp. 994–1025, 2008.

[36] G.-Y. Bae, “Neural evidence for categorical biases in working memory for location and orientation,” *bioRxiv*, 2020.

[37] W. Ma, M. Husain, and P. Bays, “Changing concepts of working memory,” *Nature neuroscience*, vol. 17, pp. 347–56, 2014.

[38] M. Cano-Colino and A. Compte, “A computational model for spatial working memory deficits in schizophrenia,” *Pharmacopsychiatry*, vol. 45 Suppl 1, S49–56, May 2012.

[39] C. Machens and C. Brody, “Design of continuous attractor networks with monotonic tuning using a symmetry principle,” *Neural computation*, vol. 20, pp. 452–85, 2008.

[40] M. Zugaro, A. Arleo, A. Berthoz, and S. Wiener, “Rapid spatial reorientation and head direction cells,” *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 23, pp. 3478–82, 2003.

[41] A. Compte, N. Brunel, P. Goldman-Rakic, and X.-J. Wang, “Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model,” *Cerebral Cortex*, vol. 10, pp. 910–923, 2000.

[42] A. Peyrache, M. Lacroix, P. Petersen, and G. Buzsáki, “Internally organized mechanisms of the head direction sense,” *Nature neuroscience*, vol. 18, pp. 569–575, 2015.

[43] E. Meyers, “Dynamic population coding and its relationship to working memory,” *Journal of Neurophysiology*, vol. 120, 2018.

[44] E. Meyers, D. Freedman, G. Kreiman, E. Mller, and T. Poggio, “Dynamic population coding of category information in inferior temporal and prefrontal cortex,” *Journal of neurophysiology*, vol. 100, pp. 1407–19, 2008.

[45] E. Spaak, K. Watanabe, S. Funahashi, and M. Stokes, “Stable and dynamic coding for working memory in primate prefrontal cortex,” *The Journal of Neuroscience*, vol. 37, pp. 3364–16, 2017.

[46] M. Stokes, M. Kusunoki, N. Sigala, H. Nili, D. Gaffan, and J. Duncan, “Dynamic coding for cognitive control in prefrontal cortex,” *Neuron*, vol. 78, pp. 1–12, 2013.

[47] M. Stokes, “’activity-silent’ working memory in prefrontal cortex: A dynamic coding framework,” *Trends in cognitive sciences*, vol. 19, 2015.

[48] O. Barak, D. Sussillo, R. Romo, M. Tsodyks, and L. Abbott, “From fixed points to chaos: Three models of delayed discrimination,” *Progress in neurobiology*, vol. 103, 2013.

[49] J. Murray, A. Bernacchia, N. Roy, C. Constantinidis, R. Romo, and X.-J. Wang, "Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 114, 2016.

[50] H. Sompolinsky, A. Crisanti, and H. Sommers, "Chaos in random neural networks," *Physical review letters*, vol. 61, 1988.

[51] K. S. Kakaria and B. L. de Bivort, "Ring attractor dynamics emerge from a spiking model of the entire protocerebral bridge," *Frontiers in Behavioral Neuroscience*, vol. 11, p. 8, 2017.

[52] J. Gámez, G. Mendoza, L. Prado, A. Betancourt, and H. Merchant, "The amplitude in periodic neural state trajectories underlies the tempo of rhythmic tapping," *PLOS Biology*, vol. 17, pp. 1–32, 2019.

[53] A. Parthasarathy, R. Herikstad, J. Bong, F. Medina, C. Libedinsky, and S.-C. Yen, "Mixed selectivity morphs population codes in prefrontal cortex," *Nature Neuroscience*, vol. 20, pp. 1770–1779, 2017.

[54] G. Elsayed, A. Lara, M. Kaufman, M. Churchland, and J. Cunningham, "Reorganization between preparatory and movement population responses in motor cortex," *Nature Communications*, vol. 7, p. 13 239, 2016.

Appendix A: Risk Assessment Retrospective

The risk assessment for this project identified that there was no increase in risk to everyday life as a result of this project, as the entire work was performed with remotely run simulations.

Appendix B: Disruption Due to COVID-19

This project did not require any in-person laboratory experiments, and was carried out entirely from home. The project-specific disruption caused by COVID-19 was negligible.