

26 Introduction

27 Recurrent neural networks (RNNs) have been employed to model computation in neurophysio-
28 logical tasks (Mante et al., 2013; Hennequin et al., 2014; Sussillo et al., 2015; Michaels et al.,
29 2016; Song et al., 2016; Chaisangmongkon et al., 2017; Miconi, 2017; Song et al., 2017). In these
30 studies, the RNN is trained to perform tasks and reproduce empirically observed behavior. Exam-
31 ples include an animal's kinematics or electromyography during a motor task or its psychometric
32 curve during a decision making task. Further, the RNN can be trained so that its artificial neurons
33 recapitulate key statistics of neurons recorded from experiments, both on the single unit and pop-
34 ulation level. Training techniques to achieve this include regularizing the network to avoid complex
35 patterns (Sussillo et al., 2015; Michaels et al., 2016), introducing architectural constraints such
36 as Dale's law (Song et al., 2016), and utilizing biologically plausible learning rules including those
37 based on reinforcement learning (Miconi, 2017; Song et al., 2017). RNNs that recapitulate the
38 behavior and key statistics of the neural population have then been analyzed to propose mech-
39 anisms for how recurrent computation occurs in cortical circuits (Sussillo & Barak, 2013; Mante
40 et al., 2013; Chaisangmongkon et al., 2017). The RNN may also generate hypotheses that can be
41 tested in future neurophysiological experiments (Chandrasekaran, 2017).

42 The existence of a diversity of training approaches that meaningfully change artificial neuron mo-
43 tifs raises several questions. For example, does the particular training approach matter? Said
44 differently, can a variety of RNNs, each trained in a different way but nevertheless all resembling
45 empirical neural activity, employ different dynamical mechanisms? If so, what are the key consid-
46 erations in using RNNs as *in silico* models of cortical circuits? We address these questions by
47 changing various design variables for RNNs and assessing how these changes affect the RNN's
48 motifs and dynamics. In particular, we vary (1) the nonlinear activation of the RNN, (2) rate reg-
49 ularization during training, and (3) task input configuration. We perform these comparisons for a
50 common motor neuroscience task: the delayed reach task. We chose this task because prior work
51 in motor systems neuroscience has proposed a concrete dynamical mechanism employed by the
52 motor cortex to perform this task (Ames et al., 2014; Churchland et al., 2006; Afshar et al., 2011;
53 Kaufman et al., 2014). Therefore, we are able to make comparisons to neurophysiological results
54 at the level of the RNN's single units, population, and dynamics.

55 We consider how design choices affect the RNN's ability to recapitulate key behavioral and neural
56 features from experiments. From this, we find that is important to recapitulate both single unit
57 and population motifs. That is, it is possible to find artificial neurons that resemble single unit
58 peri-stimulus time histograms (PSTHs) but that do not capture key population features in the neu-
59 rophysiological data. Further, we show that distinct RNNs can resemble neurophysiological data
60 while using fundamentally different dynamical mechanisms. We illustrate this idea in Fig 1b-d,
61 where for the same neural population activity evolving in two dimensions, different dynamics (de-
62 noted by the flow fields) may give rise to the same population activity. By visualizing the RNN's dy-

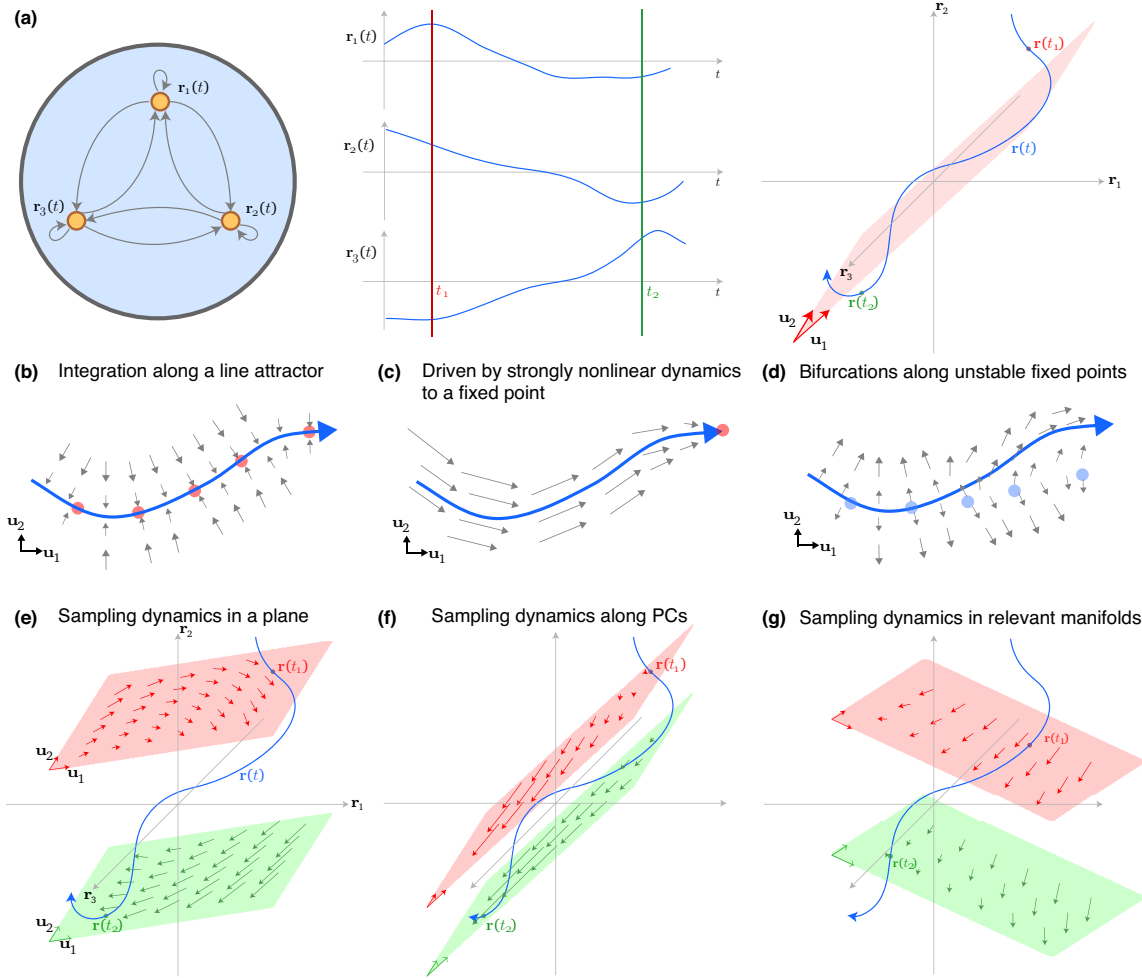


Figure 1: Illustration of sampling RNN dynamics. **(a)** Toy example of an RNN with three example units. The units' firing rates through time, $r(t)$, is plotted as a trajectory (blue) in 3 dimensions, but it largely evolves in a 2 dimensional plane indicated in red, defined by the vectors u_1 and u_2 . **(b)** The inputs may drive the trajectory slowly along a line attractor. Red dots denote stable fixed points. **(c)** The dynamics may be strong and cause the trajectory to be strongly driven to a stable fixed point. **(d)** The trajectory may be driven along regions of bifurcation, with slow unstable attractors denoted by blue dots. **(e)** For a given basis, defined by u_1 and u_2 , it is possible to project the RNN dynamics into a given plane. Here, we show a sampling rule where the values in orthogonal dimensions are set to the trajectory values. An obstacle is that the trajectories sampled at two different times, t_1 and t_2 , may have very different dynamics, indicated by the flow field arrows in the red and green planes. **(f)** If the dynamics are relatively smooth, one strategy to address this obstacle is to ensure the sampling planes, shown in red and green, are close to each other. This is achieved by sampling the principal components. **(g)** Another approach is to sample dynamics in "dynamics relevant" manifolds, where the views of the dynamics may not change as drastically depending on the sampling.

63 dynamical equations, we demonstrate that RNN input design can substantially modify the network's
 64 dynamical mechanisms. Finally, we explore consequences of computation using these distinct
 65 dynamical mechanisms, including robustness to noise and generalization to new tasks.

66 **Materials and Methods**

67 **Description of RNN and training**

68 An RNN is composed of N artificial neurons (or units) that receive input from N_{in} time-varying
69 inputs $\mathbf{u}(t)$ and produce N_{out} time-varying outputs $\mathbf{z}(t)$. The RNN defines a network state, given
70 by $\mathbf{x}(t) \in \mathbb{R}^N$; the i th element of $\mathbf{x}(t)$ is a scalar describing the “currents” of the i th artificial neuron.
71 The network state is transformed into the artificial neuron firing rates (or network rates) through
72 the transformation:

$$\mathbf{r}(t) = f(\mathbf{x}(t)), \quad (1)$$

73 where $f(\cdot)$ is an activation function applied elementwise to $\mathbf{x}(t)$. The activation function is typically
74 nonlinear, endowing the RNN with nonlinear dynamics and expressive power. In this work, we use
75 $f(x) = \tanh(x)$ as well as $f(x) = \max(x, 0)$, also known as the rectified linear unit or $\text{relu}(\cdot)$. In the
76 absence of noise, the continuous time RNN is described by the equation

$$\tau \dot{\mathbf{x}}(t) = -\mathbf{x}(t) + \mathbf{W}_{\text{rec}} \mathbf{r}(t) + \mathbf{W}_{\text{in}} \mathbf{u}(t) + \mathbf{b}_{\text{rec}}, \quad (2)$$

77 where τ is a time-constant of the network, $\mathbf{W}_{\text{rec}} \in \mathbb{R}^{N \times N}$ defines how the artificial neurons are
78 currently connected, $\mathbf{b}_{\text{rec}} \in \mathbb{R}^N$ defines a constant bias, and $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N \times N_{\text{in}}}$ maps the RNN inputs
79 onto each neuron. We note that equation 1 can also be used to calculate the dynamics of the
80 network rates, $\dot{\mathbf{r}}(t)$. This quantity is useful because it describes how the network rates evolve
81 through time. In neurophysiological studies, this is equivalent to calculating the dynamics of the
82 recorded neuron firing rates (Churchland et al., 2012; Kao et al., 2015).

83 The output of the network is given by a linear readout of the network rates, i.e.,

$$\mathbf{z}(t) = \mathbf{W}_{\text{out}} \mathbf{r}(t), \quad (3)$$

84 where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_{\text{out}} \times N}$ maps the network rates onto the network outputs. We trained the RNN
85 to minimize the mean-square error between its output, $\mathbf{z}(t)$, and a desired output, $\mathbf{z}_{\text{des}}(t)$. In
86 addition to this objective, we included several regularizations to improve training. In particular,
87 we regularized the L_2 -norm (Euclidean norm) of \mathbf{W}_{in} , \mathbf{W}_{rec} , and \mathbf{W}_{out} to penalize larger weights.
88 We also regularized the L_2 -norm of $\mathbf{r}(t)$ across all time, as was done in (Sussillo et al., 2015;
89 Michaels et al., 2016) to penalize larger rates, which are not encountered in biological neurons
90 due to their refractory period. We later report that this regularization has an important impact
91 on firing rates during movement preparation. Finally, we incorporated gradient clipping and the
92 regularization proposed by Pascanu and colleagues to ameliorate vanishing gradients (Pascanu
93 et al., 2012). Training was performed using stochastic gradient descent, with gradients calculated
94 using backpropagation through time. For gradient descent, we used the Adam optimizer, which

95 is a first order optimizer incorporating adaptive gradients and momentum (Kingma & Ba, 2014).
96 Finally, because reaching behavior is highly stereotyped, we allowed training to continue until
97 the coefficient of determination in kinematic reconstruction on validation data exceeded $R^2 >$
98 0.997.

99 Visualizing RNN dynamics

100 The RNN's dynamics are fully described by equation 2. Thus, one may qualitatively assess the
101 RNN's dynamical mechanism by visualizing this equation. However, in most scenarios, the RNN
102 is composed of a relatively large number of neurons, N (e.g., typically $N > 100$). By treating each
103 artificial neuron as an independent dimension, this implies that $\dot{\mathbf{r}}(t)$ is N -dimensional, and thus
104 not trivial to visualize. One way to address this problem is to consider that, in many scenarios, not
105 unlike what is observed in neural population activity in motor cortex (Yu et al., 2009; Cunningham &
106 Yu, 2014), the dimensionality of the N artificial neurons are correlated and can thus be adequately
107 described in a D dimensional subspace, where $D < N$. Hence, while the dynamics implemented
108 by the RNN cannot be visualized in a straightforward manner if N is large, it may be possible to
109 do so if the dynamics can be appropriately sampled in $D = 1$ to 3 dimensions.

110 There are important considerations in visualizing dynamics in a low-dimensional subspace. The
111 primary consideration is that the dynamics in any D -dimensional projection will differ depending
112 on the activity in the remaining $N - D$ dimensions. Hence, depending on the values the network
113 rates take on in the remaining $N - D$ dimensions, the visualized dynamics may differ in a minor
114 or substantial way. To illustrate this concept, consider the 3D example shown in Fig 1a. As shown
115 in Fig 1a, we can measure the firing rates of each artificial neuron for a given input, and plot the
116 trajectory of $\mathbf{r}(t)$ in 3 dimensions, where each dimension is defined by the activity of one artificial
117 neuron.

118 In Fig 1e, we introduce the notion of projecting the RNN dynamics into a given subspace. Consider
119 an orthonormal basis given by $\mathbf{U}_3 = [\mathbf{u}_1 \ \mathbf{u}_2 \ \mathbf{u}_3]$ where each $\mathbf{u}_i \in \mathbb{R}^N$ and $\mathbf{U}_3^T \mathbf{U}_3 = \mathbf{I}$. We can
120 define a two-dimensional trajectory by projecting the network rates into the subspace spanned by
121 $\mathbf{U}_2 = [\mathbf{u}_1 \ \mathbf{u}_2]$. The low-dimensional trajectory in this subspace is given by

$$122 \quad \mathbf{s}(t) = \mathbf{U}_2^T (\mathbf{r}(t) - \mu), \quad (4)$$

122 where μ is the mean of $\mathbf{r}(t)$ across time, and its dynamics can be calculated as

$$\dot{\mathbf{s}}(t) = \mathbf{U}_2^T \dot{\mathbf{r}}(t) \quad (5)$$

$$= \mathbf{U}_2^T \frac{f(\mathbf{x}(t) + \dot{\mathbf{x}}(t)\Delta t) - f(\mathbf{x}(t))}{\Delta t}. \quad (6)$$

123 However, this sampling rule is naïve in the following way. In Fig 1e, we consider the trajectory at

124 times t_1 and t_2 , denoted by $\mathbf{r}(t_1)$ in red and $\mathbf{r}(t_2)$ in green, respectively. If the trajectory $\mathbf{r}(t)$ has a
125 relatively large component along \mathbf{u}_3 , the dynamics may be very different (e.g., at time t_2 , illustrated
126 by the green plane). This is because:

$$\dot{\mathbf{r}}(t) = \mathbf{U}_2 \dot{\mathbf{s}}(t) + \mathbf{u}_3 \mathbf{u}_3^T \dot{\mathbf{r}}(t),$$

127 and thus, the low-dimensional dynamics embedded in the high-dimensional space (given by $\mathbf{U}_2 \dot{\mathbf{s}}(t)$,
128 and plotted as the flow field trajectories in Fig 1e) may change if $\mathbf{u}_3^T \dot{\mathbf{r}}(t)$ is large. We note that it is
129 possible to sample the dynamics accurately at any single time point t , since the component of $\mathbf{r}(t)$
130 in the orthogonal complement of \mathbf{U}_D is known. Hence, it is possible to visualize local dynamics
131 over time in a movie by re-sampling the low-dimensional dynamics at every time t for a given \mathbf{U}_D ;
132 such a movie is shown in Supplementary Movie 1.

133 To address the changing dynamics, we propose two heuristics to find the low-dimensional sub-
134 space, \mathbf{U}_D (where D is the number of dimensions) to sample RNN dynamics. We wish to
135 find a matrix \mathbf{U}_D with D orthonormal columns so that $\mathbf{U}_D^T \mathbf{U}_D = \mathbf{I}$. \mathbf{U}_D defines the subspace
136 where we will visualize the network rates $\mathbf{r}(t)$ as well as their dynamics $\dot{\mathbf{r}}(t)$. With this definition,
137 $\mathbf{P}_D = \mathbf{U}_D \mathbf{U}_D^T$ is a projector matrix into the subspace spanned by \mathbf{U}_D . To create a meaningful
138 dynamical portrait from which it may be possible to glean intuition as to how the RNN performs
139 a given task, the subspace should capture meaningful variance in the data, *as well as* capture a
140 faithful view of the dynamics. We enumerate two heuristics to sample these dynamics:

- 141 1. Intuitively, the components of $\mathbf{r}(t)$ along the remaining $N - D$ dimensions do not change
142 dramatically. In this manner, the sampled D -dimensional subspace is approximately the
143 same across time. In the context of Fig 1f, this corresponds to the red and green subspaces
144 being relatively close to each other. Assuming a smoothness in the RNN dynamics, if this
145 separation is sufficiently small, the dynamics will not change drastically. This smoothness
146 assumption is valid for the `tanh` nonlinearity, but for the `relu` nonlinearity fails at $\mathbf{x}(t) = 0$.
147 This projection has the added benefit of finding the projection that maximizes the projected
148 data variability $\mathbf{U}_D^T \mathbf{r}(t)$. Formally, this projection can be found by maximizing the variance of
149 $\mathbf{U}_D^T \mathbf{r}(t)$. The solution of this optimization is called principal components analysis, where \mathbf{U}_D
150 correspond to the first D left singular vectors of $[\mathbf{r}(1) \ \mathbf{r}(2) \ \dots \ \mathbf{r}(T)]$, with T being the horizon
151 of the data.
- 152 2. Intuitively, the projected dynamics in “dynamics relevant” dimensions ought to be oriented in
153 similar directions over the course of the epoch. This reflects that the dynamics will not vary
154 substantially over the course of the epoch, as illustrated in Fig 1g. This may be achieved by
155 optimization of an appropriate loss function over Stiefel manifolds.

156 In this work, we found that heuristic 1, projection along principal components, was sufficient for our
157 analyses. While we also performed an optimization under heuristic 2, we found that this did not
158 affect any conclusions. After finding \mathbf{U}_D , we visualized the network rates and their dynamics by

159 using equations 4 and 5 (and substituting U_D for U_2), respectively. Finally, we also visualized the
160 fixed points of the RNN's dynamics using the approach presented by Sussillo and Barak (Sussillo
161 & Barak, 2013).

162 **RNN task and training**

163 We trained the RNN on a variant of the delay task used by Ames and colleagues (Ames et al.,
164 2014). This study proposed a specific dynamical mechanism that we sought to probe with RNNs.
165 In this task, a monkey is instructed to hold a center target. After holding the center target for
166 700 – 1100 ms, a peripheral target is cued in one of eight locations uniformly spaced on a circle,
167 45° apart beginning at 0° . The monkey continues to hold the center target while planning to reach
168 to the cued target. After a random delay period, ranging from 0 – 900 ms, the monkey is given a
169 go cue and is allowed to perform a reach to the prompted target. Upon reaching the target, the
170 monkey then holds on the target for a 500 ms hold time to successfully acquire the target, ending
171 the trial. This task is diagrammed in Fig 2a.

172 The RNN inputs were the target's x -position, the target's y -position, and a go cue signal. Because
173 we were interested in assessing the effect of inputs on dynamical mechanisms, we trained with two
174 different go cue configurations. In the “sustained RNN” (Fig 2b, orange) the go cue was encoded
175 with a sustained signal indicating whether to withhold movement (signal high) or not (signal low)
176 as in prior studies (Sussillo et al., 2015; Michaels et al., 2016). In the “pulsed RNN” (Fig 2b, blue)
177 the go cue was encoded as a transient pulse indicating that movement could occur. This go pulse
178 could correspond to when the animal is cued that he may move by a transient cue (e.g., a brief
179 and temporary visual cue). This pulse may also be interpreted as reflecting that the state of the
180 task has changed, so that the animal may now make a reach, analogous to a signal that triggers
181 movement (Erlhagen & Schöner, 2002; Kaufman et al., 2016). An additional motivation for using
182 the pulse is that prior tasks have used transient cues, such as networks trained to process a
183 transient movement instruction (Hennequin et al., 2014). The RNN transformed these inputs into
184 four outputs: the x - and y -positions of the hand, and the x - and y -velocities of the hand. In this
185 manner, the input was 3-dimensional, $\mathbf{u}(t) \in \mathbb{R}^3$, and the output was 4-dimensional, $\mathbf{z}(t) \in \mathbb{R}^4$.
186 Our trained networks had 100 artificial neurons, so that $\mathbf{x}(t), \mathbf{r}(t) \in \mathbb{R}^{100}$.

187 Like in the study by Ames and colleagues, we trained the network with reaches having delay
188 periods ranging from 0 – 900 ms after a 700 – 1100 ms center-hold period. After each delay period,
189 we had the network produce a reach following a fixed reaction time of 150 ms. After the reach
190 transient, the network was then trained to generate zero x - and y -velocities and appropriate final
191 positions for a hold period. Instead of a static 500 ms hold period used by Ames and colleagues,
192 we allowed the hold period to be from any length from 500 to 1500 ms, so that the network didn't
193 learn specific timings (i.e., to use a region of slow dynamics for only 500 ms). We trained the
194 network to produce reaches to 8 targets uniformly spaced on a circle. The targets were 45° apart

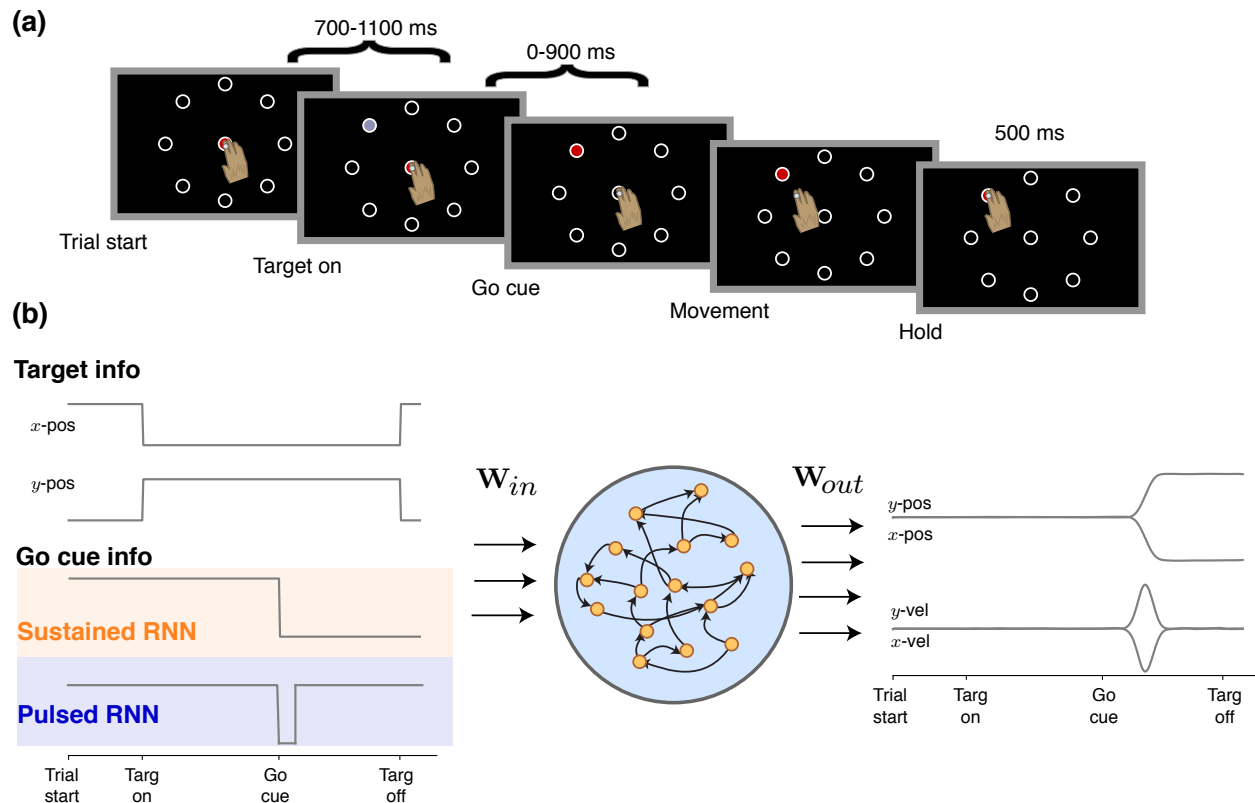


Figure 2: Delayed reach task and RNN training. **(a)** Schematic of the RNN task used by Ames and colleagues. For RNN training, the hold time was allowed to be variable for anywhere from 500 to 1500 ms so that the RNN did not learn specific timings. **(b)** Example (and representative) target inputs and outputs of the RNN. The RNN was trained until it achieved a coefficient of determination, $R^2 > 0.997$, on the task. The RNN outputs both position and the velocity. The sustained RNN encodes the go cue as a signal to withhold movement (1) or to allow movement (0) while the pulsed RNN encodes the go cue as a transient cue that indicates a change in the state of the task (i.e., that the monkey can now execute his planned reach).

195 beginning at 0° . In addition to these delayed reaches, on 10% of trials, we introduced “catch trials”
 196 to the RNN where a target may not have appeared, or if it appeared, the go cue was never given.
 197 In both instances, the network had to sustain zero output.

198 In the task by Ames and colleagues, there were also occasional *switch* trials, where the target
 199 was switched on 20% of trials. Following this target switch, the monkey was given a second delay
 200 period ranging from 0 – 900 ms followed before the go cue was delivered. We explicitly did not train
 201 on this task because we were interested in assessing how the RNN would generalize to it.

202 Results

203 Before delving into design choices, we found that it was possible to train an RNN to recapitulate
 204 key features of the neural activity in a delayed reach task, as reported in prior studies (Sussillo
 205 et al., 2015; Michaels et al., 2016). The hyperparameters for this network are listed in Supplemen-

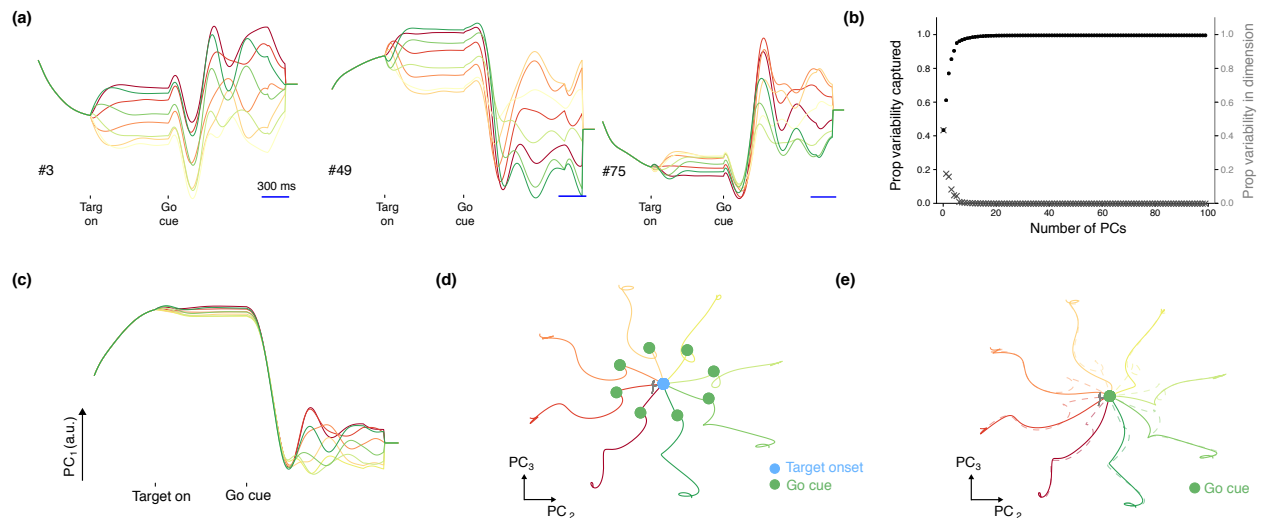


Figure 3: Sample PSTHs and population trajectories of the RNN. **(a)** Example PSTHs of artificial neurons in the RNN, where each color denotes one of the eight possible reach conditions. The PSTHs achieve a stable firing rate in the preparatory period, followed by multiphasic activity during perimovement. The condition with the highest firing rate may also change through time. **(b)** Proportion of variance captured by the principal components. 5 PCs capture 90.8% of the PSTH variability, and 10 PCs 98.2% of the variability. **(c)** PC_1 , capturing 43.7% of the signal variance, demonstrates properties consistent with the condition independent signal proposed by Kaufman and colleagues. It is the largest response component of the RNN rates, but largely does not reflect movement type (until well after the go cue has been delivered). **(d)** RNN rate trajectories in the PC_2 and PC_3 axis with a delay period. For each reach condition, the trajectories reach a stable set of neural states prior to go cue delivery. These are indicated by the trajectory locations at the green dot. Adjacent conditions are topographically organized (0° : red, 45° : dark orange, 90° : orange, 135° : light orange, 180° : yellow, 225° : light green, 270° : green, 315° : dark green). The gray part of the trajectory represents the baseline activity. **(e)** Trajectories (in bold) when the RNN performs the task without a delay period. This shows that the preparatory neural states are not obligatory, consistent with the findings of Ames and colleagues.

206 tary Table 1. Fig 3a shows peristimulus time histograms (PSTHs) of artificial neurons for delayed
 207 reaches to eight different directions. These PSTHs plateau during the delay period (Tanji & Evarts,
 208 1976; Weinrich et al., 1984; Churchland et al., 2010; Sussillo et al., 2015; Michaels et al., 2016),
 209 have substantial heterogeneity and multiphasic activity during perimovement (Sergio et al., 2005;
 210 Churchland & Shenoy, 2007; Churchland et al., 2010), and change preferred directions over time
 211 (Churchland & Shenoy, 2007; Michaels et al., 2016), reflected by the fact that the condition with the
 212 highest firing rate is not the same across the entire trial. Fig 3b-e shows that the artificial neural
 213 population also recapitulates qualitative observations from neurophysiologically recorded neural
 214 populations. We found that only 5 PCs were required to capture more than 90% of the PSTH vari-
 215 ability, as shown in Fig 3b, demonstrating that the population is low-dimensional (Yu et al., 2009;
 216 Cunningham & Yu, 2014; Ames et al., 2014; Sadtler et al., 2014; Kao et al., 2015; Gallego et al.,
 217 2017; Gao & Ganguli, 2015), that PC_1 , capturing 43.7% of the PSTH variability, strongly resembled
 218 a high variance condition-independent signal (Kaufman et al., 2016) (Fig 3c), that artificial neural
 219 population activity had topographic organization in the PCs (Santhanam et al., 2009) (Fig 3d), and
 220 that the neural population achieved a prepare-and-hold state attractor (Churchland et al., 2006)
 221 (Fig 3d) but that this attractor was not obligatory (Ames et al., 2014) (Fig 3e).

222 **Rate regularization and activation function affect preparatory activity**

223 Prior studies using RNNs to model motor cortex use the hyperbolic tangent (\tanh) activation func-
224 tion (Michaels et al., 2016; Sussillo et al., 2015) or a variant of it (Hennequin et al., 2014). Recent
225 studies have also used the rectified linear unit (relu) nonlinearity to model various decision making
226 tasks (Song et al., 2016, 2017). We note the relu nonlinearity has proliferated in several engi-
227 neering applications, in part due to the faster training times and that the gradient of the relu is
228 either zero or one, which is favorable for backpropagation (Krizhevsky et al., 2012; Szegedy et al.,
229 2015; He et al., 2016). We found that the choice of activation function impacts preparatory activity
230 during the delay period. Preparatory activity captures a substantial proportion of neural variability.
231 Typically, preparatory activity plateaus to a stable level prior to movement onset (Tanji & Evarts,
232 1976; Weinrich et al., 1984; Churchland et al., 2010; Sussillo et al., 2015; Michaels et al., 2016). In
233 a dynamical systems framework, the population preparatory activity evolves to a subspace called
234 the “prepare-and-hold” state that is beneficial for the upcoming reach (Churchland et al., 2006;
235 Afshar et al., 2011; Ames et al., 2014).

236 Given its prior use in RNN models of motor cortex, we first considered the hyperbolic tangent non-
237 linearity. Interestingly, we found that rate regularization (weighted by λ_r) was important for achiev-
238 ing preparatory activity that was qualitatively consistent with neurophysiological data. When rate
239 regularization was relatively small, we found that artificial neurons in the RNN had little prepara-
240 tory activity (Fig 4a, leftmost panel). This can be observed by recognizing that population activity
241 at the time of the go cue essentially overlapped with population activity at target onset. This so-
242 lution is not unreasonable because the RNN’s outputs remain zero during both the baseline and
243 preparatory epochs. While target information is available to the RNN in the preparatory period, it
244 does not necessarily have to act (i.e., change its state) upon this information until the go cue is
245 given. To this end, the RNN can delay processing target information until the go cue is given and
246 still successfully perform the task.

247 We found that, for the hyperbolic tangent nonlinearity, increasing rate regularization increased
248 the amount of preparatory activity in the network. This is shown for several values of λ_r in the
249 remaining panels of Fig 4a, and is summarized by Fig 4b, which shows the ratio of lengths between
250 the preparatory trajectory and the movement trajectory. The trajectory ratios are calculated in the
251 high-dimensional artificial neuron activity space and not in the low-dimensional PCs. By observing
252 the PSTHs of the activity at different levels of regularization (Supp Fig 1), we observe that rate
253 regularization causes the rates to achieve (1) smaller overall peak values and (2) intermediate
254 activations in the preparatory epoch. In this manner, rate regularization causes the \tanh RNN
255 to have stronger preparatory dynamics, effectively partitioning computation into two segments: a
256 preparatory dynamical system (driving the activity to a fixed point, denoted by the green dots in
257 Fig 4a) followed by the movement dynamical system (trajectory after the green dot). This is most
258 apparent in the rightmost panel of Fig 4a. We note that this population activity is consistent with
259 what is qualitatively observed in motor cortex. For the hyperbolic tangent nonlinearity, this is an

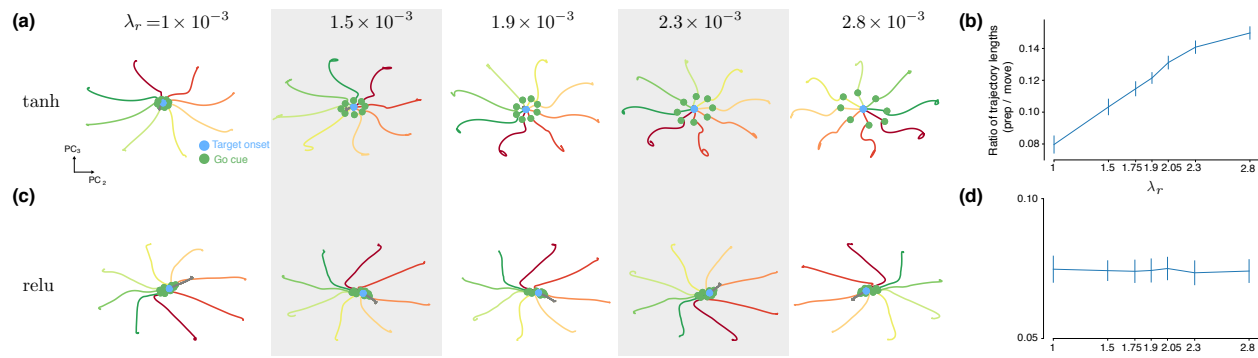


Figure 4: Rate regularization increases preparatory activity in `tanh` but not `relu` RNNs. **(a)** Population trajectories of `tanh` RNNs in PCs 2 and 3 (analogous to Fig 3d) for different values of λ_r , which weights the L_2 norm penalty on the rates. As λ_r increases, the preparatory trajectory becomes larger relative to the movement trajectory. **(b)** Ratio of the the preparatory trajectory length (in all N dimensions, not only in selected PCs) divided by the movement trajectory length. As λ_r increases, the preparatory trajectory length becomes relatively larger. Error bars are standard deviations across 8 separate RNNs trained at each value of λ_r . **(c)** Same as (a) but for `relu`. As λ_r increases, there is not a noticeable increase in preparatory activity. **(d)** Same as (b) but for `relu`.

260 energetically favorable outcome for the network.

261 We emphasize that increasing rate regularization does not always result in more preparatory ac-
 262 tivity. In fact, when we used the `relu` activation, we observed empirically that the network finds a
 263 solution that has little preparatory activity irrespective of λ_r (Fig 4c; trajectory lengths summarized
 264 in Fig 4d). We note that this was not because rate regularization was not “active” due to other reg-
 265 ularizations dominating the optimization cost; in fact, when we removed all regularization except
 266 rate regularization, the networks still had units with very little preparatory activity across 4 orders
 267 of magnitude (from $\lambda_r = 10^{-3}$ to $\lambda_r = 10$). PSTHs of the `relu` RNN are shown in Supp Fig 2,
 268 and demonstrate that although the maximum rate may decrease as rate regularization increases,
 269 the preparatory activity does not appear to increase in variability relative to movement activity.
 270 Together, these results suggest that the `relu()` activation, for these instantiations of RNNs, does a
 271 poorer job of capturing key features in the neurophysiological activity. Further, we point out that
 272 it was not the case that units with preparatory activity were absent in the network. Rather, Supp
 273 Fig 2 demonstrates that it was possible to find `relu` units that had substantial preparatory activ-
 274 ity. Hence, when considering RNNs to model neurophysiological tasks, its important to not only
 275 find single unit examples that resemble physiological activity, but that also recapitulate population
 276 level features. For the rest of this work, we utilize the `tanh()` activation with rate regularization
 277 $\lambda_r = 1.9 \times 10^{-3}$.

278 RNN dynamics in the sustained RNN during a delayed reach task

279 We next visualized the dynamics of the RNN (see Materials and Methods) as displayed in Fig 5.
 280 We also visualized the stable attractor regions of the dynamics by using the approach of (Sus-
 281 sillo & Barak, 2013). We found that the RNN implemented a dynamical mechanism that can be

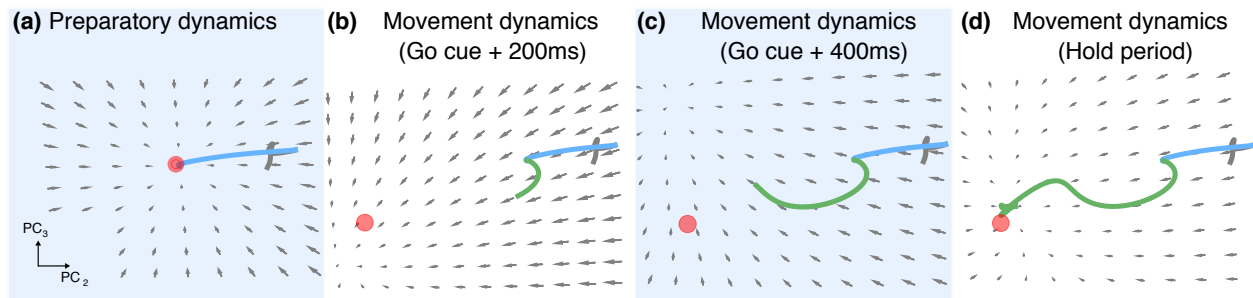


Figure 5: Composition of dynamical systems mechanism. **(a)** The delay period trajectory in PCs 2 and 3. The gray portion of the trajectory corresponds to the baseline period of the task. The blue portion of the trajectory corresponds to the delay period of the task. The state converges along essentially linear dynamics to a stable attractor (shown as a red dot). The attractor was found as the location in state space that minimized $\|\dot{x}\|^2$ below a threshold tolerance. **(b)** The movement dynamics 200 ms after go cue onset. The green portion of the trajectory corresponds to the post go cue period of the task. The dynamics are strongly driven towards a single stable attractor region. There appear to be non-zero dynamics overlapping the attractor because the trajectory is not in the same plane as the attractor (in orthogonal dimensions). **(c)** The movement dynamics 400 ms after go cue onset. The dynamics have changed due to trajectory movement in the orthogonal dimension. **(d)** Approaching the hold period, we see that the dynamics converge on the stable attractor.

282 interpreted as a composition of dynamical systems to single stable attractor regions. Key features
283 of this mechanism were proposed by Ames and colleagues to describe neural population activity
284 during a delayed reach task (Ames et al., 2014). In particular, Ames and colleagues proposed two
285 principal dynamical systems: a “preparatory” dynamical system implicated in planning a reach to
286 a desired target, and a “movement” dynamical system corresponding to the execution of the reach
287 after the go cue is delivered. The preparatory dynamical system has a stable attractor correspond-
288 ing to each prompted target. The neural state converges to this attractor, the prepare-and-hold
289 state, during the delay period. This state is a favorable initial condition for the subsequent move-
290 ment (Churchland et al., 2006; Afshar et al., 2011). When the go cue is given, the movement
291 dynamical system is engaged, driving the trajectory through a path associated with movement
292 generation.

293 By visualizing the RNN’s dynamical equations, we were able to qualitatively probe how the RNN
294 uses nonlinear dynamics to perform the task. We found that during the delay period, the RNN
295 implemented an analogous preparatory dynamical system. Upon target presentation, the trajec-
296 tories were driven by this preparatory dynamical system to a single stable attractor region as in
297 Fig 5a. This stable attractor location was target dependent. The RNN achieved different prepara-
298 tory attractors and dynamics for a given target because the network’s dynamics at a given time,
299 $\dot{x}(t)$, are modified by an input-dependent additive factor, $\mathbf{W}_{in}\mathbf{u}(t)$. Hence, each unique input can
300 be interpreted as setting up a unique dynamical system. This enables the network to, prior to the
301 go cue, instantiate different preparatory dynamics with different stable attractor regions for each
302 prompted target.

303 When the go cue was delivered, the changing input drastically modified the dynamics, so that the
304 trajectories were driven along paths associated with output generation (i.e., the movement dynam-

305 ical system). We found that this movement dynamical system was comprised of a single stable
306 attractor region. The movement dynamical system utilized strong dynamics to drive the RNN's rate
307 trajectories at relatively high speeds, as shown in Fig 5b,c. In this manner, the mechanism is not
308 integration along slow points on a line attractor (e.g., Mante et al. (2013)), but rather abrupt state
309 transitions from stable attractor to stable attractor. A video of these dynamics is shown in Supp
310 Movie 1. These types of dynamics, illustrated in Fig 1c, have been observed in another study
311 (Chaisangmongkon et al., 2017). An increase in the speed of neural trajectories following the go
312 cue is consistent with experimental observations from PMd and M1 (Afshar et al., 2011) (their Fig
313 3c). Note that when target presentation is simultaneous with the go cue so that there is no delay
314 period, the movement dynamical system is immediately engaged, and trajectories are driven by
315 the movement dynamical system to its single stable attractor region. As a result, the prepara-
316 tory dynamical system attractor is not obligatory. Because the preparatory dynamical system has
317 not been engaged for enough time, the trajectories will not achieve the preparatory attractor, a
318 phenomena also observed in neurophysiological data (Ames et al., 2014).

319 **RNNs qualitatively recapitulating neurophysiological motifs may utilize different** 320 **dynamical mechanisms**

321 We next wondered if task design considerations could produce RNNs that, while looking qualita-
322 tively similar to neurophysiological data, utilize distinct dynamical mechanisms. To this end, we
323 trained the earlier described pulsed RNN to perform a delayed reach task, using the same hyper-
324 parameters as the sustained RNN. In the training set, the pulsed go cue was delivered for 150 ms.
325 This pulse may also be interpreted as reflecting that the state of the task has changed, so that
326 the animal may now make a reach, analogous to a signal that triggers movement (Erlhagen &
327 Schöner, 2002; Kaufman et al., 2016). We are not suggesting that this pulse length would be
328 reasonable for experiments; although we chose to use 150 ms, this pulse length can be varied.
329 The RNN was capable of performing the pulsed go cue task at the same level as the single at-
330 tractor RNN (training terminated when $R^2 > 0.997$ on validation data, example output trajectory
331 shown in dark blue in Fig 6a). Its recurrent computation was similarly low-dimensional, with 5 PCs
332 capturing 91.8% of the PSTH variability (Supp Fig 3a). This RNN also bore hallmarks of neuro-
333 physiological responses, including: neural activity being organized topographically (Supp Fig 3f),
334 the trajectories achieving a “prepare-and-hold” state in the delay period (Supp Fig 3f), and that
335 these states were not obligatory (Supp Fig 3e). We do note that condition-independent variance,
336 though present, appeared to be smaller in this network, with a large proportion appearing in PC 3
337 (Supp Fig 3b-d).

338 In this task design, the input during the delay period is the same as the input during the movement
339 period post go cue. We reasoned that under the insight that each input changes the RNN's dynam-
340 ics, as seen in Fig 5 and Supp Movie 1, this RNN cannot use a composition of dynamical systems

341 with single stable attractor regions. Doing so would imply that the delay and movement periods
342 must converge to the same stable attractor, and hence the delay and movement periods would
343 converge to the same output under this mechanism. Such an RNN would be unable to adequately
344 perform this task. We note that while we have chosen a pulsed go cue, a similar conclusion holds
345 for RNNs which use transient cues, such as the networks trained by Hennequin and colleagues
346 with a movement instruction cue (Hennequin et al., 2014). Analogously, this network produced
347 two different output transients (pre-preparatory and post-movement) for the same input.

348 Although the trajectories in condition-relevant dimensions demonstrate qualitatively similar tra-
349 jectories to neurophysiological data, how does the RNN dynamically achieve this, if not by the
350 mechanism used by the sustained RNN? To answer this question, and recognizing the RNN must
351 be capable of achieving two steady-state outputs, we pulsed the go cue to determine what dura-
352 tion of go cue was required for the pulsed RNN to settle to the correct final kinematics as opposed
353 to returning to the preparatory state kinematics (i.e., zero positions and velocities). We delivered
354 different input pulses, as shown in Fig 5a, and found that when the pulsed go cue was delivered
355 for less than 85 ms, the pulsed RNN produced transient kinematics that decayed back to zero.
356 However, when the go cue was delivered for greater than 85 ms, the pulsed RNN produced the
357 correct final kinematic output corresponding to the prompted reach.

358 By visualizing the trajectories and dynamics, as shown in Fig 6b,c, it is clear that the pulsed
359 RNN sets up at least two stable attractor regions. Our optimization did not identify any other
360 attractors, suggesting that the pulsed RNN implements a bistable dynamical system. One stable
361 state is associated with the preparatory period, analogous to the prior RNN, where the RNN rates
362 converge to during the delay period. The second stable state is the region associated with making
363 a movement to the prompted target. By visualizing the dynamics, shown in Fig 6c, we were able
364 to identify bifurcation dynamics associated with the task. These bifurcation dynamics could be
365 viewed in the projections of PCs 2 and 3. The go cue pulse drives the trajectory to this region of
366 bifurcation, and depending on the trajectory's location, it will either settle back to the preparatory
367 state (with zero kinematic output, i.e., left of the illustrated bifurcation axis) or settle to the state
368 associated with the correct kinematic output (i.e., right of the illustrated bifurcation axis).

369 These results demonstrate that, for trajectories having qualitative similarities to neurophysiologi-
370 cally observed data, different dynamics may be at play. While both RNNs utilize a composition of
371 dynamical systems (Ames et al., 2014), we found that the employed mechanism differed substan-
372 tially from task input design (i.e., strongly driving trajectories to single attractors versus implement-
373 ing a region of bifurcation). In considering how to then militate between these two mechanisms,
374 we asked which mechanism was more robust to input noise, as could occur from suboptimal pro-
375 cessing of the task inputs. To this end, we added independent zero-mean Gaussian noise to the
376 inputs, and assessed the RNN's performance as a function of the standard deviation of the Gaus-
377 sian noise. We found that increasing the input noise affects the network in at least two distinct
378 ways.

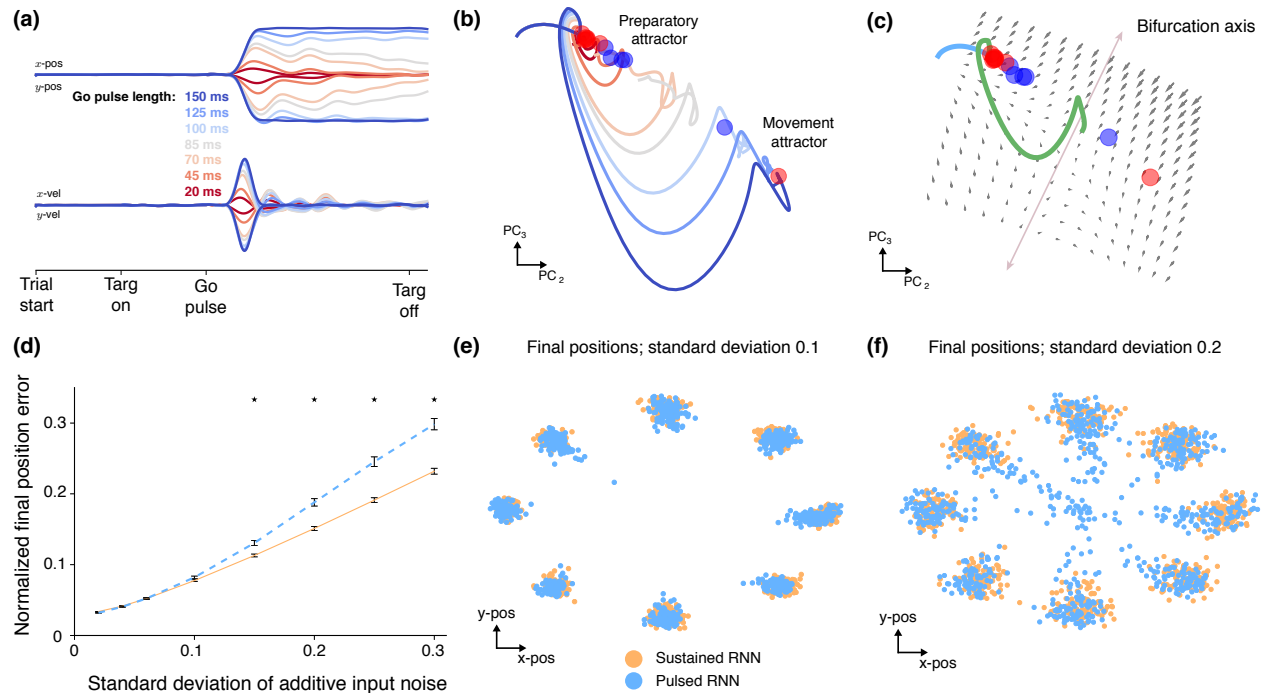


Figure 6: An RNN trained to perform a delayed reach task with a transient go cue. **(a)** Example output for an RNN that was trained on a pulsed go cue task to make a reach to the target at 315° . The output is shown for pulsing the go cue at different lengths, denoted by different colors. When the go cue is pulsed for greater than 85 ms, the RNN eventually outputs the correct final x and y -positions. When the go cue is pulsed for 85 ms or less, the RNN decays to output incorrect final zero x - and y -positions. **(b)** Neural trajectories for the reach to the target at 315° for different length go pulses. The trajectories either decay back to the preparatory state (left attractor region) or eventually converge to the stable attractor associated with movement generation (right attractor region). Red circles denote stable slow regions of state space; blue circles denote unstable slow regions of state space. **(c)** The dynamics at the point of bifurcation. The bifurcation axis is illustrated in light purple. Left of the axis, the dynamics will drive the trajectory to decay back to the preparatory attractor. Right of the axis, the dynamics will eventually drive the trajectory to the attractor associated with the correct final output. **(d)** The y-axis denotes the normalized final position error (normalized so that the final position is 1). The x-axis denotes the standard deviation of independent zero mean Gaussian noise added to the inputs. The dotted line represents the performance of the pulsed RNN, while the solid line represents the performance of the sustained RNN. As input noise increases, the pulsed RNN has worse final position performance. Stars denote significant differences in the mean (bootstrap, 1001 shuffles, $p < 0.01$). **(e)** Final positions to the eight targets for RNNs of both mechanisms when the input noise standard deviation is 0.1. Each dot represents the final position on a single trial. Both RNNs still generate relatively accurate outputs. **(f)** When the input noise standard deviation is increased to 0.2, the pulsed RNN has several trials where the final position begins to decay back to the center target, which is the kinematic output corresponding to the preparatory attractor. The hold time was increased to 2000 ms to show this slow decay. Trials which end at intermediate locations may reflect trajectories in slow regions of decay back to the preparatory attractor, as well as variable end points due to large input noise.

379 First, for both RNNs, because the stable attractor region is input-dependent, noisier inputs cause
 380 the stable attractor region to be variable, resulting in greater neural trajectory end point variabil-
 381 ity, and hence, kinematic end point variability. We observed this effect, as end point deviation
 382 increased with the standard deviation of Gaussian noise (Fig 6d) and in Fig 6e, substantial vari-
 383 ability can be observed in the end positions. Interestingly, however, we did not observe a significant
 384 difference in performance between the sustained and pulsed RNNs when the standard deviation
 385 was less than or equal to $\sigma = 0.1$, which is approximately 10% of the input signal ($p = 0.69$, boot-
 386 strap with 1001 shuffles). Hence, the effect of input noise varying attractor location was similar in

387 both networks.

388 Second, we found that for the pulsed RNN, input noise may cause the trajectory to not cross the
389 bifurcation axis, resulting in eventual relaxation to the preparatory attractor. As shown in Fig 6d,
390 when the standard deviation increased beyond $\sigma = 0.15$, the pulsed RNN had worse final end
391 point performance than the single attractor RNN ($p < 0.01$, bootstrap with 1001 shuffles). Impor-
392 tantly, for the pulsed RNN, we observed that its state occasionally relaxed back to the preparatory
393 attractor, corresponding to a $(0, 0)$ output center position, as can be observed by the final end
394 point of the kinematics shown in Fig 6f. This demonstrates that in the presence of noise, RNN task
395 performance will similarly degrade until the point where noise causes the pulsed RNN's state to
396 converge to the incorrect attractor. This suggests that, for the purposes of performing a delayed
397 reach task, the sustained RNN is more robust under input noise.

398 **RNN generalization to new tasks**

399 Finally, we wondered the extent to which a qualitative understanding of the RNN's dynamics could
400 inform task generalization. We believe this is an important line of questioning for future RNN
401 studies. In particular, by constraining what tasks the RNN is trained on, it is possible to comment
402 on what dynamical mechanisms are sufficient to carry out certain tasks. As an example, can
403 an RNN using the composition of preparatory and movement dynamical systems generalize to
404 perform a target switch task? The target switch task is diagrammed in Fig 7a. We hypothesized
405 this ought be possible; indeed, Ames and colleagues used a qualitatively consistent dynamical
406 mechanism to describe how motor cortex performs a target switch task. By training an RNN to
407 only perform a delayed reach task and utilizing the composition of dynamical systems, we can
408 assess whether this dynamical mechanism is sufficient to enable the network to perform related
409 tasks it was not trained on. We considered three variants of a target switch task, where the target
410 switches at different times: (1) before the go cue, (2) simultaneous with the go cue, and (3) after the
411 go cue. The first two were considered in neurophysiological experiments by Ames and colleagues
412 (Ames et al., 2014).

413 Consider first the sustained RNN. When the target switch is delivered prior to the go cue, the
414 preparatory dynamical system is changed from that associated with the before-switch target to
415 that of the after-switch target. As a result, the preparatory stable attractor changes when the
416 target is switched, and the RNN's rates will converge to the single stable attractor associatd with
417 the switched target. When the go cue is then delivered, the RNN will execute the reach as it did
418 in a delayed reach task. We found this was the case, as illustrated in Fig 7b-e, mimicking the
419 experimental results presented by Ames and colleagues (Ames et al., 2014). When the go cue
420 is given simultaneously with the target switch, this is analogous to performing a delayed reach
421 from a suboptimal initial condition. The network will achieve the correct end behavior, because
422 when the go cue is delivered, it must settle to the single stable attractor region of the movement

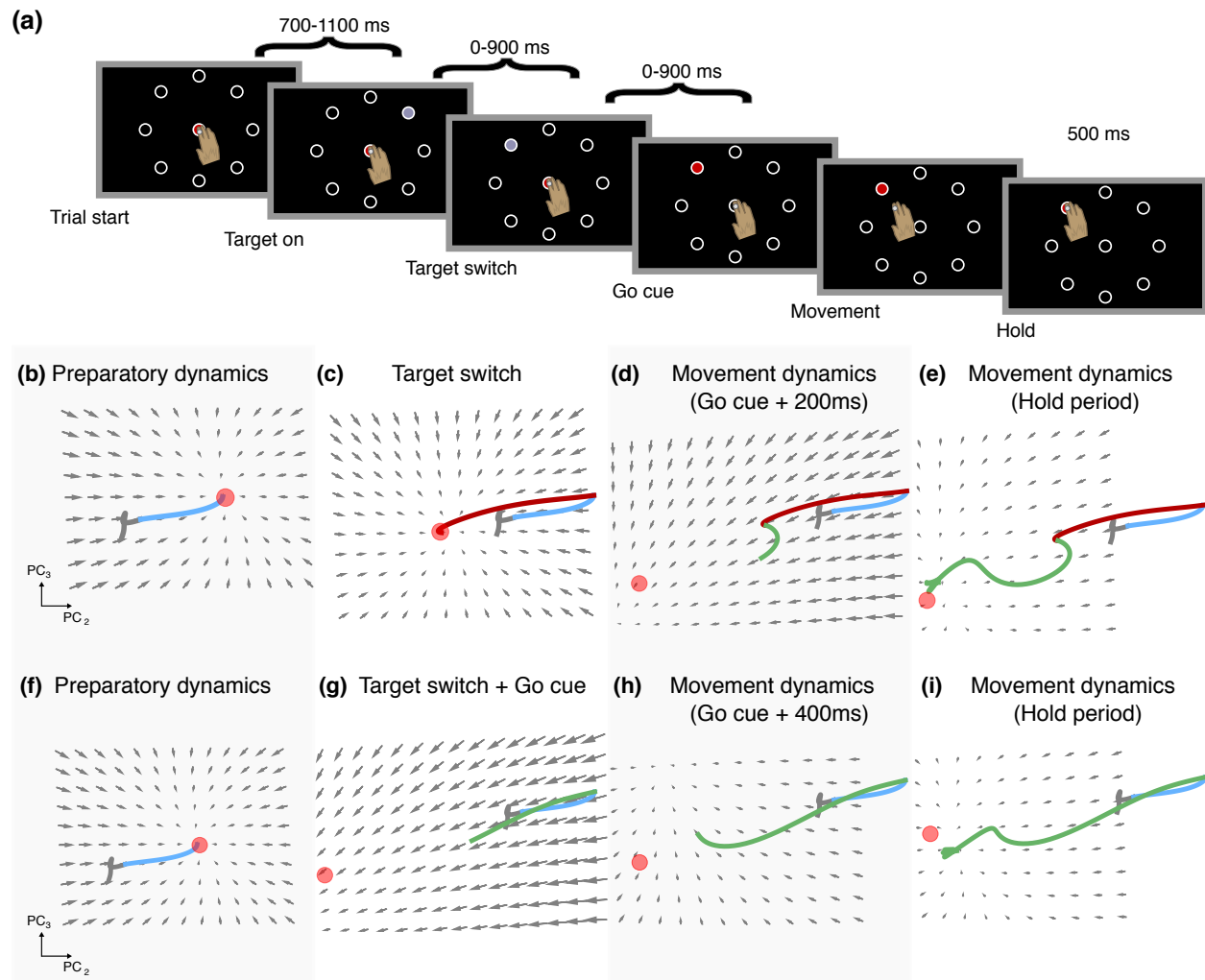


Figure 7: RNN dynamics during a target switch task. **(a)** Schematic of the target switch task when the go cue is delivered either with a delay or contemporaneously with the go cue. **(b-e)** RNN dynamics during a target switch task, with an additional delay period to re-prepare. **(b)** Preparatory dynamics during the delay period (blue trajectory). **(c)** Dynamics during the switch period (red trajectory). The new target changes the stable attractor region, and the dynamics drive the trajectory to this attractor. **(d-e)** Dynamics following the go cue, recapitulating the same trajectory seen in Fig 5c,d. **(f-i)** Same as **(b-e)** but when the go cue is given contemporaneously with the target switch.

423 dynamical system, as shown in Fig 7f-i. However, there is the potential for initial aberrant kinematic
 424 activity from initiating the movement dynamical system from the fixed point of an incorrect target
 425 plan. We found that this was not the case, and indeed the RNN was able to carry out the target
 426 switch task with no additional delay period, illustrated by representative examples in Fig 8a. Thus,
 427 the mechanism used by the RNN to perform the delayed reach task generalizes to perform two
 428 variants of the target switch task, even though it was not trained on it. Analogous arguments apply
 429 to the pulsed RNN. We found that, like the sustained RNN, the pulsed RNN also generalized to
 430 the target switch task when the switch was delivered either at the same time or preceding the go
 431 cue, as shown in Fig 7a.

432 We also considered a task when the target is switched after the go cue. This task comprises an

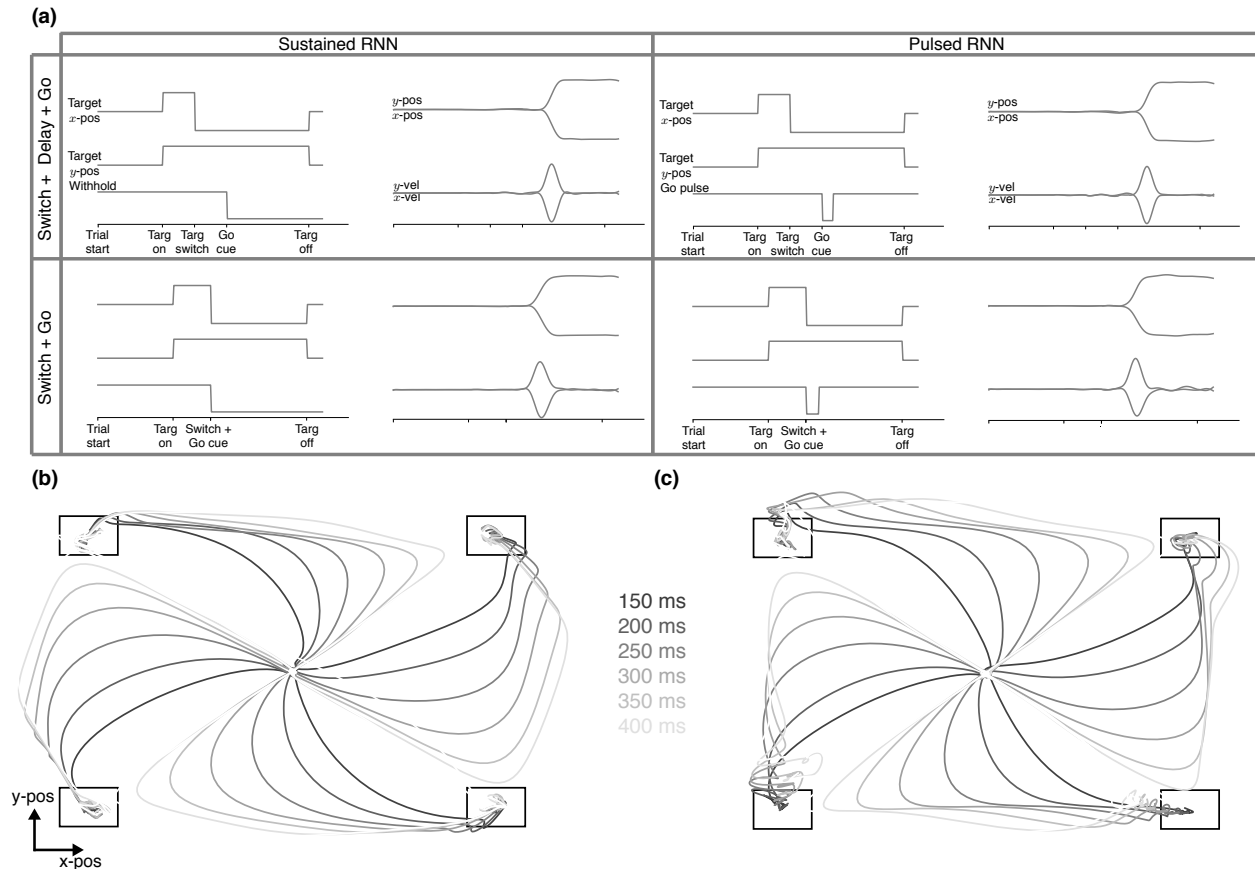


Figure 8: Example behavior for the RNNs generalizing to perform the target switch task. **(a)** Both RNNs discussed generalize to the target switch task when the target switch occurred either before or at the same time at the go cue. Both RNNs had a coefficient of determination $R^2 > 0.99$ in reconstructing the output kinematics despite not being trained on the target switch task explicitly. **(b)** RNN output positions during a target switch task, where the switch is delivered after the go cue. Here, we show four target switch conditions, when switches occur to adjacent targets. This output corresponds to the RNN trained to perform the sustained go cue delayed reach task. The different shades correspond to output reach trajectories when the target was switched some amount of time following the go cue (legend in between panels (a) and (b)). **(c)** Same as **(b)** but for the pulsed RNN trained on the pulsed go cue delayed reach task.

433 online corrective component, where feedback of the arm's kinematics play an important role in
 434 updating motor commands to reach to a new target. While we did not account for this corrective
 435 component, we nevertheless found that the open-loop RNN could reasonably perform the task
 436 as the correct target input dictated the stable attractor location. This is shown in Fig 8b,c, where
 437 for different lengths of time after the go cue, a target switch is delivered in the middle of the trial.
 438 We found that even though there was not a corrective feedback component and the target was
 439 abruptly changed, the RNN made a smooth and reasonable trajectory between targets.

440 We found that the pulsed RNN had poorer generalization in the presence of input and recurrent
 441 noise. We incorporated input noise and recurrent noise into RNNs as they performed a task where
 442 the target switched 200 ms after the go cue. We found that, in general, the pulsed RNN had poorer
 443 robustness to both input noise and recurrent noise (Fig 9a,b) across varying levels of noise. In

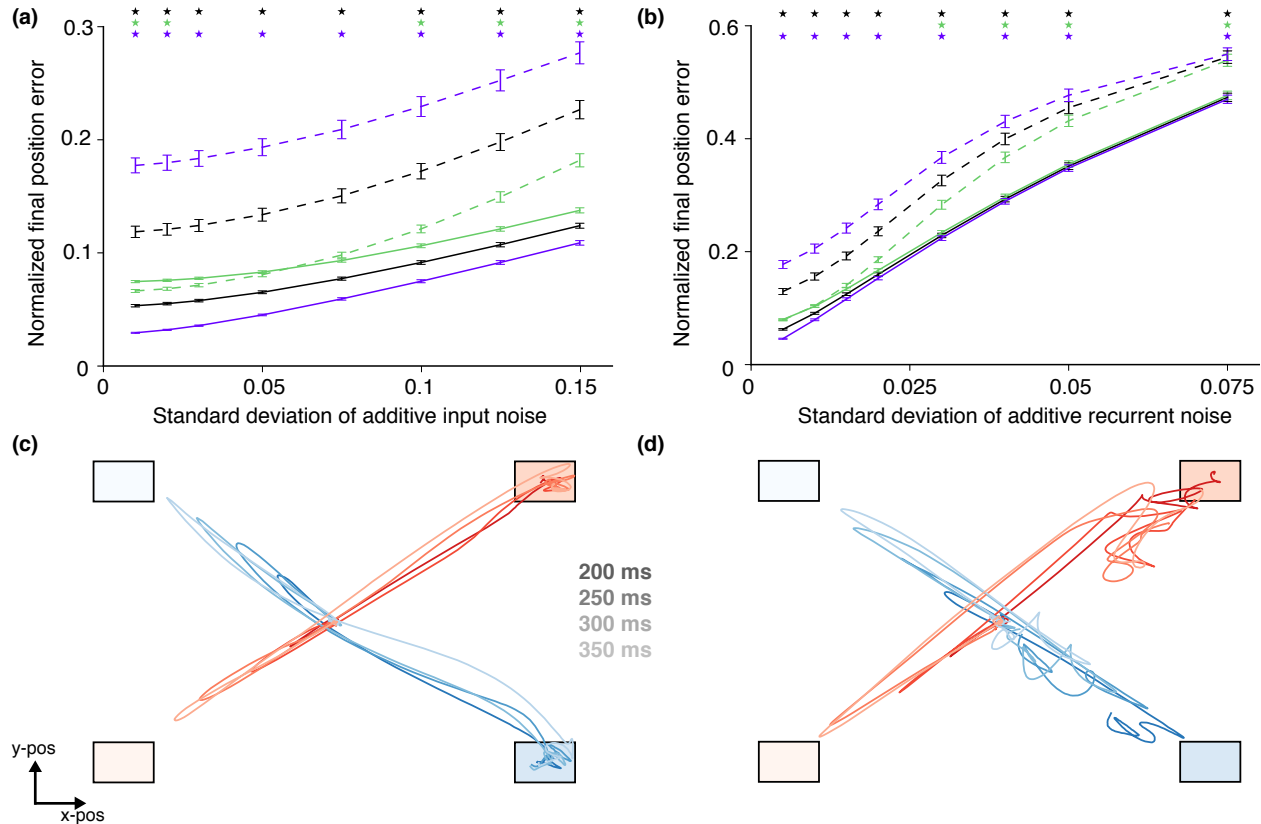


Figure 9: **(a)** Normalized error in the final position as a function of the standard deviation of Gaussian noise added to the input. The final position was measured at 1200 ms after go cue onset, when the trials were terminated. The position errors are normalized so that the target position has a distance of 1. The solid lines correspond to the performance of the sustained RNN trained on the delayed reach task when performing the target switch after go cue task, and the dotted lines correspond to the performance of the pulsed RNN. The black lines denote the error across all switch conditions. The purple lines denote the error for switch conditions where the switched target was 180° away from the pre-switch target, and the green lines correspond to the error for switch conditions where the switched target was $+90^\circ$ away from the pre-switch target. Stars denote a significant difference in the means at the level $p < 0.01$ (bootstrap, 1001 shuffles). Error bars are standard error of the mean. In general, the pulsed RNN has poorer robustness under additive input noise. **(b)** The same as **(a)** but for recurrent noise added to each artificial neuron. In general, the pulsed RNN has poorer robustness under additive recurrent noise. **(c)** Example output kinematics of the sustained RNN for target switch trials for two conditions, where the target switch is diagonal. The lighter target corresponds to the initially prompted target, and the darker target corresponds to the switched target. To make the task harder, noise was injected into the inputs. The RNN arrives at the correct final behavior, as would be expected by its dynamical mechanism. **(d)** Same as **(c)** but for the pulsed RNN trained to perform the pulsed go cue delayed reach task. One can observe that the RNN fails to perform the task adequately, achieving the incorrect final position.

444 particular, we found that the pulsed RNN especially performed worse when the target switch was
 445 maximal at 180° (purple lines in Fig 9a,b). The sustained RNN adequately performed this task,
 446 being able to make diagonal corrections, as shown in Fig 9c. However, we found that the pulsed
 447 RNN was not able to consistently perform this task (example trajectories in Fig 9d). One reason
 448 for poorer performance is that on several occasions, the output trajectories began to correct in the
 449 right direction, but do not cross the bifurcation axis, settling back to the preparatory state attractor.
 450 These results suggest that, while both the sustained and pulsed RNNs are capable of using their
 451 dynamical mechanisms to generalize to target switch tasks, the sustained RNN has more robust

452 generalization in the presence of noise.

453 Discussion

454 RNNs, being an *in silico* circuit model that is fully observed, are a promising tool to investigate
455 mechanisms underlying motor behavior (Sussillo et al., 2015; Hennequin et al., 2014; Michaels
456 et al., 2016), decision-making (Mante et al., 2013; Chaisangmongkon et al., 2017; Song et al.,
457 2017), and other neurophysiological tasks (Song et al., 2016). However, there are important con-
458 siderations when using RNNs to make conclusions about cortical computation. Our results high-
459 light two key points: (1) that it is not sufficient to demonstrate that RNNs have artificial neurons that
460 only recapitulate key motifs in single neurons, and (2) that even networks that capture both single
461 neuron and population motifs in the data may use fundamentally distinct mechanisms.

462 First, we found that it is insufficient to have RNNs merely recapitulate single neuron motifs alone.
463 In addition to single neuron motifs, RNNs ought capture population level motifs in the data. This is
464 clear in the choice of activation function to train RNNs. In RNNs trained with the `relu` activation, it
465 was possible to find artificial neurons with preparatory activity, but these RNNs did not capture key
466 population features from the data. Rather, preparatory trajectories were relatively short compared
467 to movement trajectories (Fig 4c), which is inconsistent with empirical results. Further, even for the
468 `tanh` activation function, we found that it was important to regularize the network appropriately to
469 capture preparatory variability in the population. This would not have been straightforward if only
470 looking at single neuron comparisons.

471 Second, we saw that two distinct RNNs could both recapitulate key hallmarks of neurophysiological
472 activity, but do so in fundamentally different ways. In this manner, even if an RNN recapitulates
473 both single neuron and population level motifs, careful consideration should be given to how the
474 RNN dynamically performs the task. Our results showed that by varying how the task inputs are
475 designed, the RNN can use distinct mechanisms with important consequences on generalization
476 in the presence of noise. Our results also demonstrate that in addition to regularizations (e.g.,
477 (Sussillo et al., 2015)), architectures (e.g., (Song et al., 2016)), and training rules (e.g., (Miconi,
478 2017; Song et al., 2017)), task input design can have an important effect on how the RNN's
479 computations are performed. It may be possible for future experiments to be designed in such
480 a way to militate between these mechanisms by assessing behavior in the presence of noisy
481 inputs.

482 As task design can substantially affect the dynamical mechanisms employed by the RNN, so too
483 may other hyperparameters and training paradigms. It will be appropriate to consider how RNN
484 architectures and parameters affect dynamical mechanisms, their robustness to noise, and gen-
485 eralization. For example, Song and colleagues demonstrated that it is possible to design RNNs
486 that obey biological constraints such as Dale's law (Song et al., 2016). They demonstrated that in

487 these networks, one could find single artificial neurons consistent with empirical recordings. Sim-
488 ilarly, Miconi and colleagues demonstrated that networks can be trained with biological learning
489 rules (Miconi, 2017). These learning rules, which reproduce neurophysiological features of the
490 data, may affect the network dynamics. Assessing the extent to which important features of the
491 employed dynamical mechanisms change through introducing biological constraints and learning
492 may play an important role in proposing mechanisms for cortical computation and making concrete
493 predictions for future experiments.

494 Interrogating an RNN's dynamics also has consequences for what type of dynamics may be *suf-*
495 *ficient* to carry out a class of tasks. In our work, we found that an RNN trained to only perform a
496 delayed reach task was capable of generalizing to target switch tasks, even though it wasn't trained
497 on these tasks. This shows that the mechanism employed by the RNN to perform a delayed reach
498 task endows the network with the capability of performing the target switch task. An interesting
499 line of future work may assess how the RNN's dynamics change as it is trained to perform a wider
500 assortment of tasks (Yang et al., 2017). This may describe how many, and what classes, of tasks
501 are necessary to provide an RNN with the capability of performing a different set of tasks. Further,
502 in so far as the capability to perform a variety of tasks changes the dynamical mechanisms of the
503 network, this may help to narrow the set of plausible mechanisms used to perform a given task.
504 For example, if we trained the networks in this work to perform motor tasks with perturbations
505 to the arm (e.g., Omrani et al. (2014); Nashed et al. (2012)), does the network cease to employ
506 a bistable mechanism? Another line of inquiry is to visualize the dynamics of RNNs that fail to
507 generalize, and determine what deficiencies result in poor generalization.

508 **References**

- 509 Afshar, A., Santhanam, G., Yu, B. M., Ryu, S. I., Sahani, M., & Shenoy, K. V. (2011). Single-trial
510 neural correlates of arm movement preparation. *Neuron*, 71, 555–564.
- 511 Ames, K. C., Ryu, S. I., & Shenoy, K. V. (2014). Neural dynamics of reaching following incorrect or
512 absent motor preparation. *Neuron*, 81, 438–451.
- 513 Chaisangmongkon, W., Swaminathan, S. K., Freedman, D. J., & Wang, X. J. (2017). Comput-
514 ing by robust transience: how the fronto-parietal network performs sequential, category-based
515 decisions. *Neuron*, 93, 1504–1517.e4.
- 516 Chandrasekaran, C. (2017). Computational principles and models of multisensory integration.
517 *Curr. Opin. Neurobiol.*, 43, 25–34.
- 518 Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Foster, J. D., Nuyujukian, P., Ryu, S. I., &
519 Shenoy, K. V. (2012). Neural population dynamics during reaching. *Nature*, 487, 51–56.

- 520 Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Ryu, S. I., & Shenoy, K. V. (2010). Cortical
521 preparatory activity: Representation of movement or first cog in a dynamical machine? *Neuron*,
522 68, 387–400.
- 523 Churchland, M. M. & Shenoy, K. V. (2007). Temporal complexity and heterogeneity of single-
524 neuron activity in premotor and motor cortex. *J. Neurophysiol.*, 97, 4235–4257.
- 525 Churchland, M. M., Yu, B. M., Ryu, S. I., Santhanam, G., & Shenoy, K. V. (2006). Neural variability
526 in premotor cortex provides a signature of motor preparation. *Journal of Neuroscience*, 26,
527 3697–3712.
- 528 Cunningham, J. P. & Yu, B. M. (2014). Dimensionality reduction for large-scale neural recordings.
529 *Nat. Neurosci.*, 17, 1500–1509.
- 530 Erlhagen, W. & Schöner, G. (2002). Dynamic field theory of movement preparation. *Psychol. Rev.*,
531 109, 545–572.
- 532 Gallego, J. A., Perich, M. G., Miller, L. E., & Solla, S. A. (2017). Neural manifolds for the control of
533 movement. *Neuron*, 94, 978–984.
- 534 Gao, P. & Ganguli, S. (2015). On simplicity and complexity in the brave new world of large-scale
535 neuroscience. *Curr. Opin. Neurobiol.*, 32, 148–155.
- 536 He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In
537 *Computer Vision – ECCV 2016, Lecture Notes in Computer Science*, pp. 630–645. (Springer,
538 Cham).
- 539 Hennequin, G., Vogels, T. P., & Gerstner, W. (2014). Optimal control of transient dynamics in
540 balanced networks supports generation of complex movements. *Neuron*, 82, 1394–1406.
- 541 Kao, J. C., Nuyujukian, P., Ryu, S. I., Churchland, M. M., Cunningham, J. P., & Shenoy, K. V.
542 (2015). Single-trial dynamics of motor cortex and their applications to brain-machine interfaces.
543 *Nat. Commun.*, 6, 1–12.
- 544 Kaufman, M. T., Churchland, M. M., Ryu, S. I., & Shenoy, K. V. (2014). Cortical activity in the null
545 space: permitting preparation without movement. *Nat. Neurosci.*, 17, 440–448.
- 546 Kaufman, M. T., Seely, J. S., Sussillo, D., Ryu, S. I., Shenoy, K. V., & Churchland, M. M. (2016).
547 The largest response component in motor cortex reflects movement timing but not movement
548 type. *eNeuro*, 3, ENEURO.0085–16.2016.
- 549 Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization.
- 550 Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolu-
551 tional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira,
552 C. J. C. Burges, L. Bottou, & K. Q. Weinberger, eds. (Curran Associates, Inc.), pp. 1097–1105.

- 553 Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation
554 by recurrent dynamics in prefrontal cortex. *Nature*, 503, 78–84.
- 555 Michaels, J. A., Dann, B., & Scherberger, H. (2016). Neural population dynamics during reaching
556 are better explained by a dynamical system than representational tuning. *PLoS Comput. Biol.*,
557 12, e1005175.
- 558 Miconi, T. (2017). Biologically plausible learning in recurrent neural networks reproduces neural
559 dynamics observed during cognitive tasks. *Elife*, 6.
- 560 Nashed, J. Y., Crevecoeur, F., & Scott, S. H. (2012). Influence of the behavioral goal and environ-
561 mental obstacles on rapid feedback responses. *J. Neurophysiol.*, 108, 999–1009.
- 562 Omrani, M., Pruszynski, J. A., Murnaghan, C. D., & Scott, S. H. (2014). Perturbation-evoked
563 responses in primary motor cortex are modulated by behavioral context. *J. Neurophysiol.*, 112,
564 2985–3000.
- 565 Pascanu, R., Mikolov, T., & Bengio, Y. (2012). On the difficulty of training recurrent neural networks.
566 *Proceedings of The 30th International Conference on Machine Learning*, pp. 1310–1318.
- 567 Sadtler, P. T., Quick, K. M., Golub, M. D., Chase, S. M., Ryu, S. I., Tyler-Kabara, E. C., Yu, B. M.,
568 & Batista, A. P. (2014). Neural constraints on learning. *Nature*, 512, 423–426.
- 569 Santhanam, G., Yu, B. M., Gilja, V., Ryu, S. I., Afshar, A., Sahani, M., & Shenoy, K. V. (2009).
570 Factor-analysis methods for higher-performance neural prostheses. *J. Neurophysiol.*, 102,
571 1315–1330.
- 572 Sergio, L. E., Hamel-Pâquet, C., & Kalaska, J. F. (2005). Motor cortex neural correlates of output
573 kinematics and kinetics during isometric-force and arm-reaching tasks. *J. Neurophysiol.*, 94,
574 2353–2378.
- 575 Song, H. F., Yang, G. R., & Wang, X. J. (2016). Training excitatory-inhibitory recurrent neural
576 networks for cognitive tasks: a simple and flexible framework. *PLoS Comput. Biol.*, 12, 1–30.
- 577 Song, H. F., Yang, G. R., & Wang, X.-J. (2017). Reward-based training of recurrent neural networks
578 for cognitive and value-based tasks. *Elife*, 6.
- 579 Stavisky, S. D., Kao, J. C., Ryu, S. I., & Shenoy, K. V. (2017). Motor cortical visuomotor feedback
580 activity is initially isolated from downstream targets in Output-Null neural state space dimen-
581 sions. *Neuron*, 95, 195–208.e9.
- 582 Sussillo, D. & Barak, O. (2013). Opening the black box: low-dimensional dynamics in high-
583 dimensional recurrent neural networks. *Neural Comput.*, 25, 626–649.
- 584 Sussillo, D., Churchland, M. M., Kaufman, M. T., & Shenoy, K. V. (2015). A neural network that
585 finds a naturalistic solution for the production of muscle activity. *Nat. Neurosci.*, 18, 1025–1033.

- 586 Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., &
587 Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference
588 on computer vision and pattern recognition, pp. 1–9.
- 589 Tanji, J. & Evarts, E. V. (1976). Anticipatory activity of motor cortex neurons in relation to direction
590 of an intended movement. *J. Neurophysiol.*, 39, 1062–1068.
- 591 Weinrich, M., Wise, S. P., & Mauritz, K. H. (1984). A neurophysiological study of the premotor
592 cortex in the rhesus monkey. *Brain*, 107 (Pt 2), 385–414.
- 593 Yang, G. R., Francis Song, H., Newsome, W. T., & Wang, X.-J. (2017). Clustering and composi-
594 tionality of task representations in a neural network trained to perform many cognitive tasks.
- 595 Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., & Shenoy, K. V. (2009). Gaussian-
596 process factor analysis for low-dimensional single-trial analysis of neural population activity. *J.*
597 *Neurophysiol.*, 102, 612–635.

598 **Acknowledgments**

599 We thank Chandramouli Chandrasekaran and Krishna Shenoy for helpful discussions. To train
600 RNNs, we used `pycog` (<https://github.com/xjwanglab/pycog/>) with modifications for task de-
601 sign, architecture and optimization.

602 **Author contributions**

603 JCK was responsible for design, analyses, simulation experiments, and manuscript writeup.

604 **Supplementary materials**

605 **Supplementary Movie 1.** Movie demonstrating visualized dynamics at each point in time as the
 606 RNN performs a delayed reach task. The stable fixed point jitters in the preparatory period due to
 607 numerical precision; we performed the optimization to find the fixed point at each time step. A link
 608 to this movie is at: https://seas.ucla.edu/~kao/vid/18rnn_SuppMovie1.mp4

Supplementary Table 1.

RNN parameters	Values
Number of units	100
Time constant (τ)	50 ms
Discretization bin width	10 ms
L_2 regularizer for \mathbf{W}_{in}	1×10^{-3}
L_2 regularizer for \mathbf{W}_{out}	1×10^{-3}
L_2 regularizer for \mathbf{W}_{rec}	1×10^{-3}
L_2 regularizer for $\mathbf{r}(t)$	1.9×10^{-3}
λ_Ω regularizer	2
Activation function	$\tanh(\cdot)$
Initial learning rate (Adam)	1×10^{-4}
Maximum gradient norm	0.2

Table S1: Parameters used for RNN training. The λ_Ω regularizer is described further in the study by (Pascanu et al., 2012) as well as (Song et al., 2016).

609

610 **Supplementary Figures.**

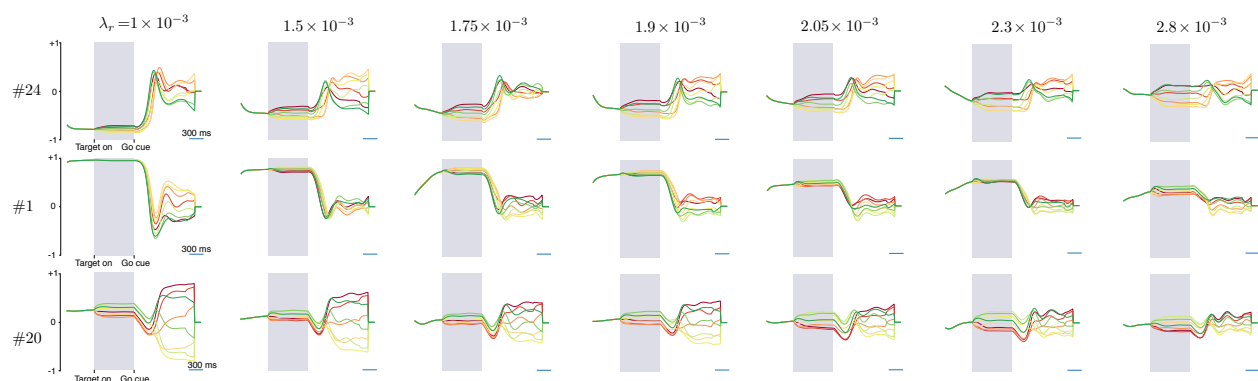


Figure S1: PSTHs for the same neuron in an RNN across seven different levels of regularization for a \tanh RNN. The region highlighted in gray corresponds to preparatory activity. The neuron is the “same” across all networks in the sense that we initialized the networks in the exact same way, with the same random seed, and they only differed in the amount of rate regularization. We found that each unit across the different RNNs shared similar motifs under this training process. In general, as rate regularization increases, the units have more preparatory activity relative to movement activity.

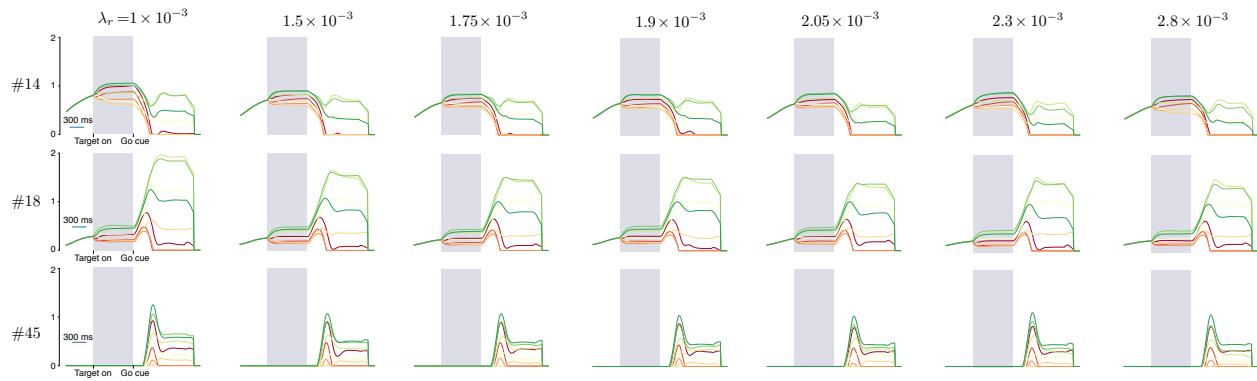


Figure S2: PSTHs for the same neuron in an RNN across 8 different levels of regularization for a `relu` RNN. The region highlighted in gray corresponds to preparatory activity. In general, even as rate regularization increases, the units have similar activity.

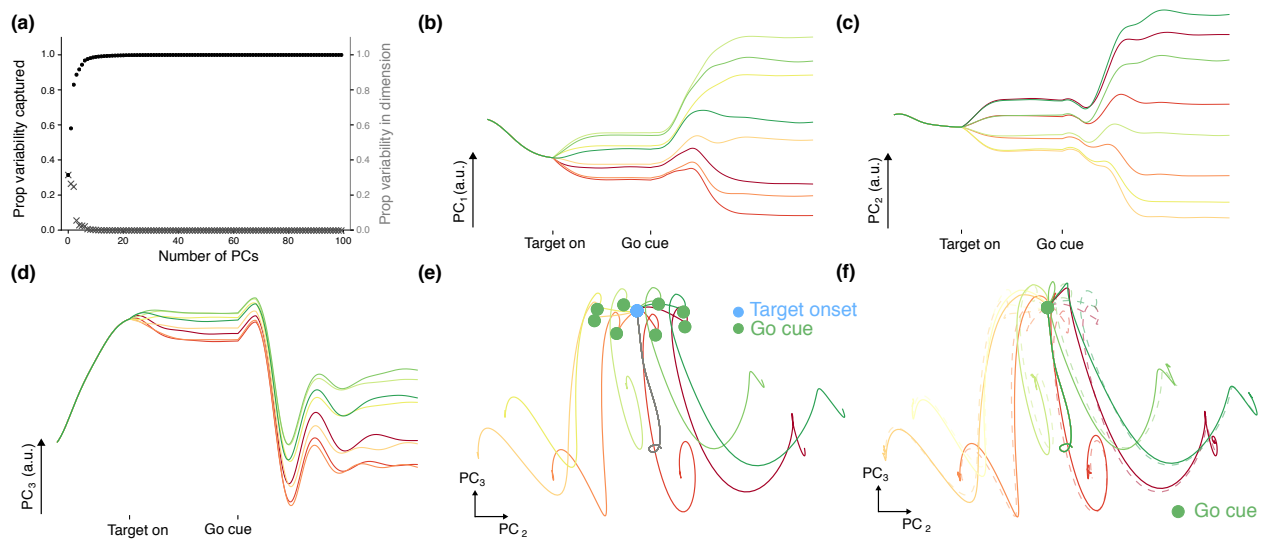


Figure S3: Principal components analysis for RNN trained to perform a delayed reach task with pulsed go cue. **(a)** Variance captured by dimension. The first 5 PCs capture 91.8% of the PSTH variability and the first 10 PCs capture 98.6% of the PSTH variability. **(b)** PC_1 of the RNN rates. PC_1 contains a substantial amount of condition dependent information. **(c)** PC_2 of the RNN rates. **(d)** PC_3 of the RNN rates. This dimension captures a large transient signal that is largely condition independent after the go cue. **(e)** Projection of PC_2 and PC_3 with a delay period. The trajectories in the delay period reach a target dependent stable region in state space, and subsequently are strongly driven along trajectories associated with movement production. **(f)** Projection of PC_2 and PC_3 shows that the delay period is not obligatory. The dotted traces are trajectories with a delay period while the solid traces are trajectories without a delay period.