J Neurophysiol 122: 2504–2521, 2019.
First published October 16, 2019; doi:10.1152/jn.00467.2018.

RESEARCH ARTICLE | *Neural Circuits*

Considerations in using recurrent neural networks to probe neural dynamics

Jonathan C. Kao[1,2]

[1]*Department of Electrical and Computer Engineering, University of California, Los Angeles, California; and* [2]*Neurosciences Program, University of California, Los Angeles, California*

Submitted 9 July 2018; accepted in final form 7 October 2019

Kao JC. Considerations in using recurrent neural networks to probe neural dynamics. *J Neurophysiol* 122: 2504–2521, 2019. First published October 16, 2019; doi:10.1152/jn.00467.2018.—Recurrent neural networks (RNNs) are increasingly being used to model complex cognitive and motor tasks performed by behaving animals. RNNs are trained to reproduce animal behavior while also capturing key statistics of empirically recorded neural activity. In this manner, the RNN can be viewed as an in silico circuit whose computational elements share similar motifs with the cortical area it is modeling. Furthermore, because the RNN's governing equations and parameters are fully known, they can be analyzed to propose hypotheses for how neural populations compute. In this context, we present important considerations when using RNNs to model motor behavior in a delayed reach task. First, by varying the network's nonlinear activation and rate regularization, we show that RNNs reproducing single-neuron firing rate motifs may not adequately capture important population motifs. Second, we find that even when RNNs reproduce key neurophysiological features on both the single neuron and population levels, they can do so through distinctly different dynamical mechanisms. To distinguish between these mechanisms, we show that an RNN consistent with a previously proposed dynamical mechanism is more robust to input noise. Finally, we show that these dynamics are sufficient for the RNN to generalize to tasks it was not trained on. Together, these results emphasize important considerations when using RNN models to probe neural dynamics.

NEW & NOTEWORTHY Artificial neurons in a recurrent neural network (RNN) may resemble empirical single-unit activity but not adequately capture important features on the neural population level. Dynamics of RNNs can be visualized in low-dimensional projections to provide insight into the RNN's dynamical mechanism. RNNs trained in different ways may reproduce neurophysiological motifs but do so with distinctly different mechanisms. RNNs trained to only perform a delayed reach task can generalize to perform tasks where the target is switched or the target location is changed.

artificial neural network; motor cortex; neural computation; neural dynamics; recurrent neural network

INTRODUCTION

Recurrent neural networks (RNNs) have been employed to model computation in neurophysiological tasks (Chaisangmongkon et al. 2017; Hennequin et al. 2014; Lukashin et al. 1996; Mante et al. 2013; Michaels et al. 2016; Miconi 2017; Remington et al. 2018; Song et al. 2016, 2017; Stroud et al. 2018; Sussillo et al. 2015; Yang et al. 2019). In these studies, the RNN is trained to perform tasks and reproduce empirically observed behavior. Examples include an animal's kinematics or electromyography during a motor task or its psychometric curve during a decision-making task. Furthermore, the RNN can be trained so that its artificial neurons reproduce key statistics of neurons recorded from experiments, both on the single unit and population level. Training techniques to achieve this include regularizing the network to avoid complex patterns (Sussillo et al. 2015), introducing architectural constraints such as Dale's law (Song et al. 2016), and utilizing biologically plausible learning rules, including those based on reinforcement learning (Miconi 2017; Song et al. 2017). RNNs that reproduce the behavior and key statistics of the neural population have then been analyzed to propose mechanisms for how recurrent computation occurs in cortical circuits (Chaisangmongkon et al. 2017; Mante et al. 2013; Sussillo and Barak 2013). RNNs may also generate hypotheses that can be tested in future neurophysiological experiments (Chandrasekaran 2017).

The existence of a diversity of RNN training approaches, hyperparameters, regularizations, and architectures that meaningfully change artificial neuron motifs raises several questions. If these RNNs are to provide insight into neurophysiology, it is important to consider how design considerations affect their single-unit and population responses. For example, how do hyperparameters, architectural constraints, and input configuration affect the network's neural population activity? Can a variety of RNNs, each trained in a different way but nevertheless bearing qualitative resemblance to empirical neural activity, employ different dynamical mechanisms? By "dynamical mechanism," we are referring to dynamical properties of the RNN, including attractors (also known as equilibrium or fixed points) that change the RNN's artificial unit activations to produce a correct output (Chaisangmongkon et al. 2017; Mante et al. 2013; Remington et al. 2018; Sussillo and Barak 2013). What are key design considerations in using RNNs as in silico models of cortical circuits?

We address these questions by varying commonly encountered design variables for RNNs on motor tasks. We then assess how these changes affect the RNN's single-unit activity, population activity, and dynamics. Of the many design variables that may be considered, we vary *1*) the nonlinear activation of the RNN, *2*) rate and parameter regularization, *3*) network architecture to incorporate excitatory and inhibitory units and Dale's law, and *4*) task input configuration. We chose

Address for reprint requests and other correspondence: J. C. Kao, 56-147H Engineering IV Building, 420 Westwood Plaza, Box 951594, Los Angeles, CA, 90095 (e-mail: kao@seas.ucla.edu).

these design variables because they vary across prior literature. We perform these comparisons for RNNs trained on a common motor neuroscience task: the delayed reach task. We chose this task because prior work in motor systems neuroscience has proposed a concrete dynamical mechanism in the motor cortex for this task (Afshar et al. 2011; Ames et al. 2014; Churchland et al. 2006; Kaufman et al. 2014). Therefore, we are able to make comparisons to neurophysiological results at the level of single units, neural population activity, and dynamics.

In this work, we first make contributions in considering how design choices affect the RNN's ability to reproduce key behavioral and neural features from experiments. From this, we find that it is important to capture both single-unit and population motifs. That is, it is possible to find artificial neurons that resemble single-unit peristimulus time histograms (PSTHs) but do not capture key population features in the neurophysiological data. Furthermore, we show that distinct RNNs can resemble neurophysiological data while using fundamentally different dynamical mechanisms. We illustrate this idea in Fig. 1, *B–D*, where similar neural population activity evolving in two dimensions may arise from different dynamics (denoted by the flow fields). We visualize the RNN's dynamical equations and

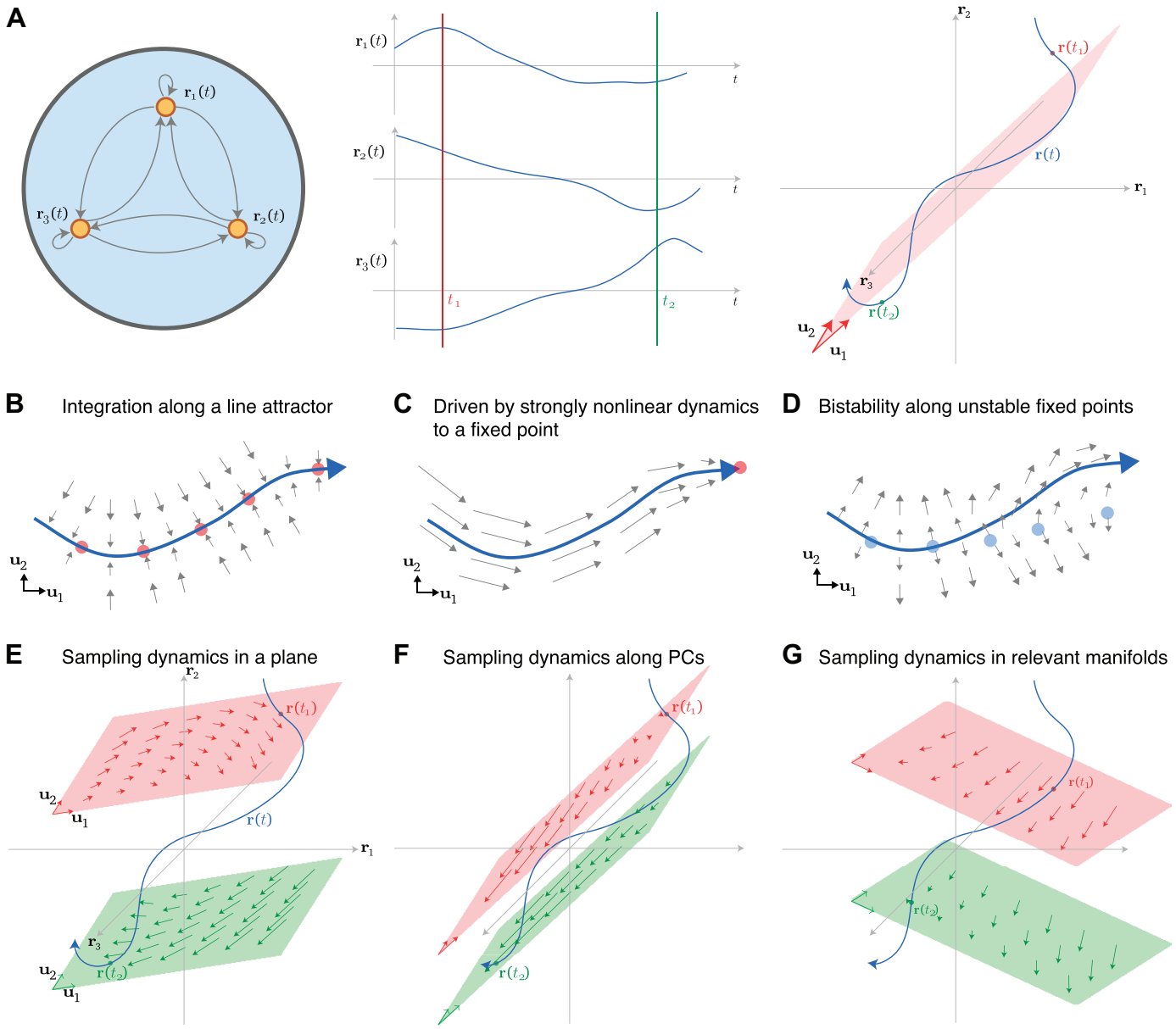

Fig. 1. Illustration of sampling recurrent neural network (RNN) dynamics. *A*: toy example of an RNN with 3 example units. The units' firing rate through time, $\mathbf{r}(t)$, is plotted as a trajectory (blue) in 3 dimensions, but it largely evolves in a 2-dimensional plane indicated in red, defined by the vectors $\mathbf{u}_1$ and $\mathbf{u}_2$. *B*: inputs may drive the trajectory slowly along a line attractor. Red dots denote stable attractors. *C*: dynamics may be strong and cause the trajectory to be strongly driven to a stable attractor. *D*: trajectory may be driven along regions of bistability, with slow unstable attractors denoted by blue dots. *E*: for a given basis, defined by $\mathbf{u}_1$ and $\mathbf{u}_2$, 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 2 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, 1 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.

attractors, finding that RNN input design can modify the network's dynamical mechanisms. A key contribution of this work is to show that two RNNs, differing only in input design, may both capture key features of empirical neural population activity but do so with distinct dynamical mechanisms. To distinguish between these distinct mechanisms, we assess the robustness and generalizability of these RNNs.

MATERIALS AND METHODS

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

$$\mathbf{r}(t) = f[\mathbf{x}(t)], \tag{1}$$

where $f(\cdot)$ is an activation function applied elementwise to $\mathbf{x}(t)$. The activation function is typically nonlinear, endowing the RNN with nonlinear dynamics and expressive capacity. In this work, we consider the activation functions:

- $f(x) = \tanh(x)$.
- $f(x) = \max(x, 0)$, also known as the rectified linear unit or relu$(\cdot)$.
- $f(x) = \max(x, \alpha x)$, with $\alpha = 0.01$ in this work. This is also known as the leaky relu.
- $f(x) = \log[1 + \exp(x)]$, also known as the softplus function, which is a smooth and differentiable function that resembles relu.

In the absence of noise, the continuous time RNN is described by the equation

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

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 recurrently 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 onto each neuron. We note that *Eq. 1* can also be used to calculate the dynamics of the network rates, $\dot{\mathbf{r}}(t)$. This quantity is useful because it describes how the network rates evolve through time. In neurophysiological studies, this is equivalent to calculating the dynamics of the recorded neuron firing rates (Churchland et al. 2012; Kao et al. 2015).

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

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

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 to minimize the mean-square error between its output, $\mathbf{z}(t)$, and a desired output, $\mathbf{z}_{\text{des}}(t)$. In addition to this objective, we included several regularizations to improve training. In particular, we regularized the L_2-norm (Euclidean norm) of $\mathbf{W}_{\text{in}}$, $\mathbf{W}_{\text{rec}}$, and $\mathbf{W}_{\text{out}}$ to penalize larger weights. We also regularized the L_2-norm of $\mathbf{r}(t)$ across all time, as was done in Michaels et al. (2016) and Sussillo et al. (2015) to penalize larger rates, which are not encountered in biological neurons due to their refractory period. We later report that this regularization has an important impact on firing rates during movement preparation. Finally, we incorporated gradient clipping and the regularization proposed by Pascanu and colleagues (2013) to ameliorate vanishing gradients. Training was performed using gradient descent, with gradients calculated using backpropagation through time. For gradient descent, we used the Adam optimizer, which is a first-order optimizer incorporating adaptive gradients and momentum (Kingma and Ba 2015). In some simulations, we also incorporated an 80:20% ratio of excitatory-to-inhibitory neurons that followed Dale's law, using the technique described in Song et al. (2016). Finally, because reaching behavior is highly stereotyped, we allowed training to continue until the coefficient of determination in kinematic reconstruction on validation data exceeded $R^2 > 0.997$.

Analyzing RNN dynamics. The RNN's dynamics are fully observed and described by *Eq. 2*. Therefore, we have an equation that precisely describes how the network's state evolves over time, enabling us to quantify dynamical properties of the RNN. In this work, we focus on quantifying the RNN's attractors, which are the network states, $\mathbf{x}$, for which $\dot{\mathbf{x}}(t) \approx 0$ for all t. That is, we aim to solve

$$-\mathbf{x} + \mathbf{W}_{\text{rec}}\mathbf{r} + \mathbf{W}_{\text{in}}\mathbf{u} + \mathbf{b}_{\text{rec}} = 0,$$

for a particular input configuration $\mathbf{u}$ and with $\mathbf{r} = f(\mathbf{x})$ (dropping the time index). RNNs may have multiple attractors; furthermore, the relationship between $\dot{\mathbf{x}}$ and $\mathbf{x}$ is nonlinear, in general precluding an analytical solution. We identify the attractors through numerical optimization over vectors $\mathbf{x}$, $\in$, $\mathbb{R}^N$ using the technique described in Sussillo and Barak (2013). That is, we identify attractors by first solving the following optimization problem:

$$\min_{\mathbf{x}} \; q(\mathbf{x}), \tag{4}$$

where we define

$$q(\mathbf{x}) = \frac{1}{2}\|\dot{\mathbf{x}}(t)\|^2$$

for convenience. We designated the minimizer, $\mathbf{x}$, of $q(\mathbf{x})$ to be an attractor if $q(\mathbf{x})$ was below a small threshold value to account for numerical considerations. This small threshold varied on RNN analysis. We trained two classes of RNNs in this work, described later. 'Sustained RNNs' utilized a threshold less than 2×10^{-8}, and 'pulsed RNNs' utilized a threshold less than 5×10^{-8}. Because these values are not zero, we do not call these fixed points but utilize the terminology 'attractor' to refer to an RNN state, characterized by slow dynamics, that the RNN converges to. For example, we later describe a preparatory RNN attractor during the delay period of the trained task. When we extended the delay period to be much longer (e.g., 100 s, two orders of magnitude longer than any delay period in training), the RNN stays at the same attractor region, and the output does not change (data not shown).

The optimization problem in *Eq. 4* is equivalent to

$$\min_{\mathbf{x}} \frac{1}{2}\|-\mathbf{x} + \mathbf{W}_{\text{rec}}\mathbf{r} + \mathbf{W}_{\text{in}}\mathbf{u} + \mathbf{b}_{\text{rec}}\|^2.$$

These attractors are calculated over epochs within a task (e.g., delay or movement epoch) under that epoch's corresponding inputs, $\mathbf{u}$. We solved this unconstrained minimization problem using a Newton conjugate gradient algorithm, also known as a truncated Newton method (Nocedal and Wright 2006). To ameliorate the curse of dimensionality in searching a large volume in $\mathbb{R}^N$, we initialized the optimization in regions of interest. These regions are in the neighborhood of $\mathbf{x}(t)$ when simulating the network with typical training inputs. We performed this optimization 1,000 times for each attractor analysis, each time initializing $\mathbf{x}$ to one of 1,000 different sampled $\mathbf{x}(t)$ PSTHs over the course of many trials. After determining an attractor, $\mathbf{x}$, we assessed whether the attractor was stable following the heuristic of Sussillo and Barak (2013). That is, we determined an attractor to be stable if, after repeated optimizations, RNN network states empirically converged to these attractor regions. We denote these stable attractors as red circles. If attractors were not stable, then we denoted them with blue circles; this could refer to saddle points or repellers.

Visualizing RNN dynamics. The RNN's dynamics are fully described by *Eq. 2*. Thus, one may qualitatively assess the RNN's dynamics by visualizing this equation and considering the RNN's attractors. However, in most scenarios, the RNN is composed of a relatively large number of neurons, N (e.g., typically $N > 100$). By

treating each artificial neuron as an independent dimension, this implies that $\dot{\mathbf{r}}(t)$ is N-dimensional and therefore not trivial to visualize. One way to address this problem is to consider that in many scenarios, not unlike what is observed in neural population activity in motor cortex (Cunningham and Yu 2014; Yu et al. 2009), the dimensionality of the N-artificial neurons is correlated and can thus be adequately described in a D-dimensional subspace, where $D < N$. Hence, although the dynamics implemented by the RNN cannot be visualized in a straightforward manner if N is large, it may be possible to do so if the dynamics can be appropriately sampled in $D = 1$–3 dimensions.

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

In Fig. 1*E*, we introduce the notion of projecting the RNN dynamics into a given subspace. Consider 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 define a two-dimensional trajectory by projecting the network rates into the subspace spanned by $\mathbf{U}_2 = [\mathbf{u}_1\ \mathbf{u}_2]$. The low-dimensional trajectory in this subspace is given by

$$\mathbf{s}(t) = \mathbf{U}_2^T\big[\mathbf{r}(t) - \mu\big], \tag{5}$$

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

$$\dot{\mathbf{s}}(t) = \mathbf{U}_2^T\dot{\mathbf{r}}(t) \tag{6}$$

$$= \mathbf{U}_2^T\frac{f\big[\mathbf{x}(t) + \dot{\mathbf{x}}(t)\,\Delta t\big] - f\big[\mathbf{x}(t)\big]}{\Delta t}. \tag{7}$$

However, this sampling rule is naïve in the following way: in Fig. 1*E*, we consider the trajectory at 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 significant variance along $\mathbf{u}_3$, the dynamics may be very different (e.g., at time t_2, illustrated by the green plane). This is because

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

thus, the low-dimensional dynamics embedded in the high-dimensional space (given by $\mathbf{U}_2\dot{s}(t)$ and plotted as the flow-field trajectories in Fig. 1*E*) may change if $\mathbf{u}_3^T\dot{r}(t)$ is large. We note that it is possible to sample the dynamics accurately at any single time point t because the component of $\mathbf{r}(t)$ in the orthogonal complement of $\mathbf{U}_D$ is known. Hence, it is possible to visualize local dynamics over time in a movie by resampling the low-dimensional dynamics at every time t for a given $\mathbf{U}_D$; such a movie is shown in Supplemental Movie S1 (see https://doi.org/10.5281/zenodo.3422566).

To address the changing dynamics, we propose two heuristics to find the low-dimensional subspace, $\mathbf{U}_D$ (where D is the number of dimensions), to sample RNN dynamics. We wish to 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 where we will visualize the network rates $\mathbf{r}(t)$ as well as their dynamics $\dot{\mathbf{r}}(t)$. With this definition,

$$\mathbf{P}_D = \mathbf{U}_D\mathbf{U}_D^T$$

is a projector matrix into the subspace spanned by $\mathbf{U}_D$. To create a meaningful dynamical portrait from which it may be possible to glean intuition as to how the RNN performs a given task, the subspace should capture meaningful variance in the data as well as a faithful view of the dynamics. We enumerate the following two heuristics to sample these dynamics:

1. Intuitively, the components of $\mathbf{r}(t)$ along the remaining $N-D$ dimensions should not change dramatically. In this manner, the sampled D-dimensional subspace is approximately the same across time. In the context of Fig. 1*F*, this corresponds to the red and green subspaces being relatively close to each other. Assuming a smoothness in the RNN dynamics, if this separation is sufficiently small, the dynamics will not change drastically. This smoothness assumption is valid for the hyperbolic tangent (tanh) and softplus nonlinearities, but not for the relu or leaky relu nonlinearity at $\mathbf{x}(t) = 0$. This projection has the added benefit of finding the projection that maximizes the projected data variability $\mathbf{U}_D^T\mathbf{r}(t)$. Formally, this projection can be found by maximizing the variance of $\mathbf{U}_D^T\mathbf{r}(t)$. The solution of this optimization is called principal components analysis, where $\mathbf{U}_D$ corresponds to the first D left singular vectors of $[\mathbf{r}(1)\ \mathbf{r}(2)\ \ldots\ \mathbf{r}(T)]$, with T being the horizon of the data.
2. Intuitively, the projected dynamics in 'dynamics relevant' dimensions ought to be oriented in similar directions over the course of the epoch. This reflects that the dynamics will not vary significantly over the course of the epoch, as illustrated in Fig. 1*G*. This may be achieved by optimization of an appropriate loss function over Stiefel manifolds. We have included one example of such an optimization to demonstrate the application of this technique, although for this work, we find heuristic 1 sufficient. In this example, in addition to minimizing a change in dynamics, we constrain the projection to capture a significant proportion of the PSTH variance. We constrained the manifold to capture at least 0.4 of the PSTH variance. For convenience, we let Σ denote the covariance matrix of the PSTHs, so that $\mathrm{Tr}(\Sigma)$ is the total PSTH variance. In optimization, we first normalized all projected dynamics, i.e., we calculated

$$\mathbf{y}(t) = \frac{\mathbf{U}_D^T\mathbf{r}(t)}{\|\mathbf{U}_D^T\dot{\mathbf{r}}(t)\|}.$$

The task we trained on has a delay, move, and hold epoch (described further below), where dynamics may differ. If we define $\mathbf{t}_{\text{delay}} = \{t\colon t \in \text{delay epoch}\}$ and analogous definitions for $\mathbf{t}_{\text{move}}$ and $\mathbf{t}_{\text{hold}}$, then the optimization problem we solved is:

$$\min_{\mathbf{U}_D}\Big[-\max_{i,j\in\mathbf{t}_{\text{delay}},i\neq j}\mathbf{y}(i)^T\mathbf{y}(j) - \max_{i,j\in\mathbf{t}_{\text{move}},i\neq j}\mathbf{y}(i)^T\mathbf{y}(j) - \max_{i,j\in\mathbf{t}_{\text{hold}},i\neq j}\mathbf{y}(i)^T\mathbf{y}(j)\Big]$$

subject to

$$\mathbf{U}_D^T\mathbf{U}_D = \mathbf{I}$$

$$\mathrm{Tr}\big(\mathbf{U}_D^T\textstyle\sum\mathbf{U}_D\big) \geq 0.4\mathrm{Tr}\big(\textstyle\sum\big)$$

This is an optimization over Stiefel manifolds, which we performed using the pymanopt toolbox (Townsend et al. 2016). We incorporated the constraint into the objective via the log-barrier method (Boyd and Vandenberghe 2004). In this manner, these dimensions minimize the largest angles between dynamics within each epoch, so that within each epoch the dynamics are similar in the projected manifolds. We show an example of this projection in Fig. 7, *E–H*.

In this work, we found that heuristic 1, projection along principal components, was sufficient for our analyses. Although we also performed an optimization under heuristic 2, we found that this did not affect any conclusions. After finding $\mathbf{U}_D$, we visualized the network rates and their dynamics by using *Eqs. 5* and *6* (and substituting $\mathbf{U}_D$ for $\mathbf{U}_2$), respectively.

We note that visualizing RNN activity in low-dimensional projections primarily provides intuition regarding the RNN's dynamics. It is

also worth noting that when we compute attractors, we do not do this in the low-dimensional space. Rather, we optimize to find attractors in the full-dimensional space and subsequently project the attractor location into the low-dimensional space to visualize it.

RNN task and training. We trained the RNN on a variant of the delay task used by Ames and colleagues (2014). This study proposed a specific dynamical mechanism that we sought to probe with RNNs. In this task, a monkey is instructed to hold a center target. After holding the center target for 700–1,100 ms, a peripheral target is cued in one of 8 locations uniformly spaced on a circle, 45° apart beginning at 0°. The monkey continues to hold the center target while planning to reach to the cued target. After a random delay period, ranging from 0−900 ms, the monkey is given a go cue and is allowed to perform a reach to the prompted target. Upon reaching the target, the monkey then holds the target for a 500-ms hold time to successfully acquire the target, ending the trial. This task is diagrammed in Fig. 2*A*.

The RNN inputs were the target's x-position, the target's y-position, and a go cue signal. Because we were interested in assessing the effect of inputs on mechanism, we trained with two different go cue configurations. In the sustained RNN (Fig. 2*B*, orange), the go cue was encoded with a sustained signal indicating whether to withhold movement (signal high) or not (signal low) as in prior studies (Michaels et al. 2016; Sussillo et al. 2015). In the pulsed RNN, the go cue was encoded as a transient pulse (decreasing from 1 to 0, then increasing back to 1, as depicted in Fig. 2*B*), indicating that movement could occur. This go pulse could correspond to when the animal is cued that he may move by a transient cue (e.g., a brief and temporary visual cue). This pulse may also be interpreted as reflecting that the state of the task has changed so that the animal may now make a reach, analogous to a signal that triggers movement (Erlhagen and Schöner 2002; Kaufman et al. 2016). An additional motivation for using the pulse is that prior tasks have used transient cues, such as networks trained to process a transient movement instruction (Hennequin et al. 2014) or a pulse (Remington et al. 2018). The RNN transformed these inputs into four outputs: the x- and y-positions and the x- and y-velocities of a desired trajectory. In this manner, the input was three-dimensional, $\mathbf{u}(t) \in \mathbb{R}^3$, and the output was four-dimensional, $\mathbf{z}(t) \in \mathbb{R}^4$. Our trained networks had 100 artificial neurons, so that $\mathbf{x}(t)$, $\mathbf{r}(t)$ are 100-dimensional vectors, $\mathbf{x}(t)$, $\mathbf{r}(t) \in \mathbb{R}^{100}$.

Like in the study by Ames and colleagues (2014), we trained the network with reaches having delay periods ranging from 0−900 ms after a 700−1,100 ms center-hold period. After each delay period, we had the network produce a reach following a fixed reaction time of 150 ms. After the reach transient, the network was then trained to generate zero x- and y-velocities and appropriate final positions for a hold period. Instead of a static 500-ms hold period used by Ames and colleagues (2014), we allowed the hold period to be from any length from 500–1,500 ms so that the network did not learn specific timings (i.e., to use a region of slow dynamics for only 500 ms). We trained the network to produce reaches to 8 targets uniformly spaced on a circle. The targets were 45° apart beginning at 0°. In addition to these delayed reaches, on 10% of trials, we introduced 'catch trials' to the RNN where a target may not have appeared, or if it appeared, the go cue was never given. In both instances, the network had to sustain zero output.

In the task by Ames and colleagues (2014), there were also occasional switch trials, where the target was switched on 20% of

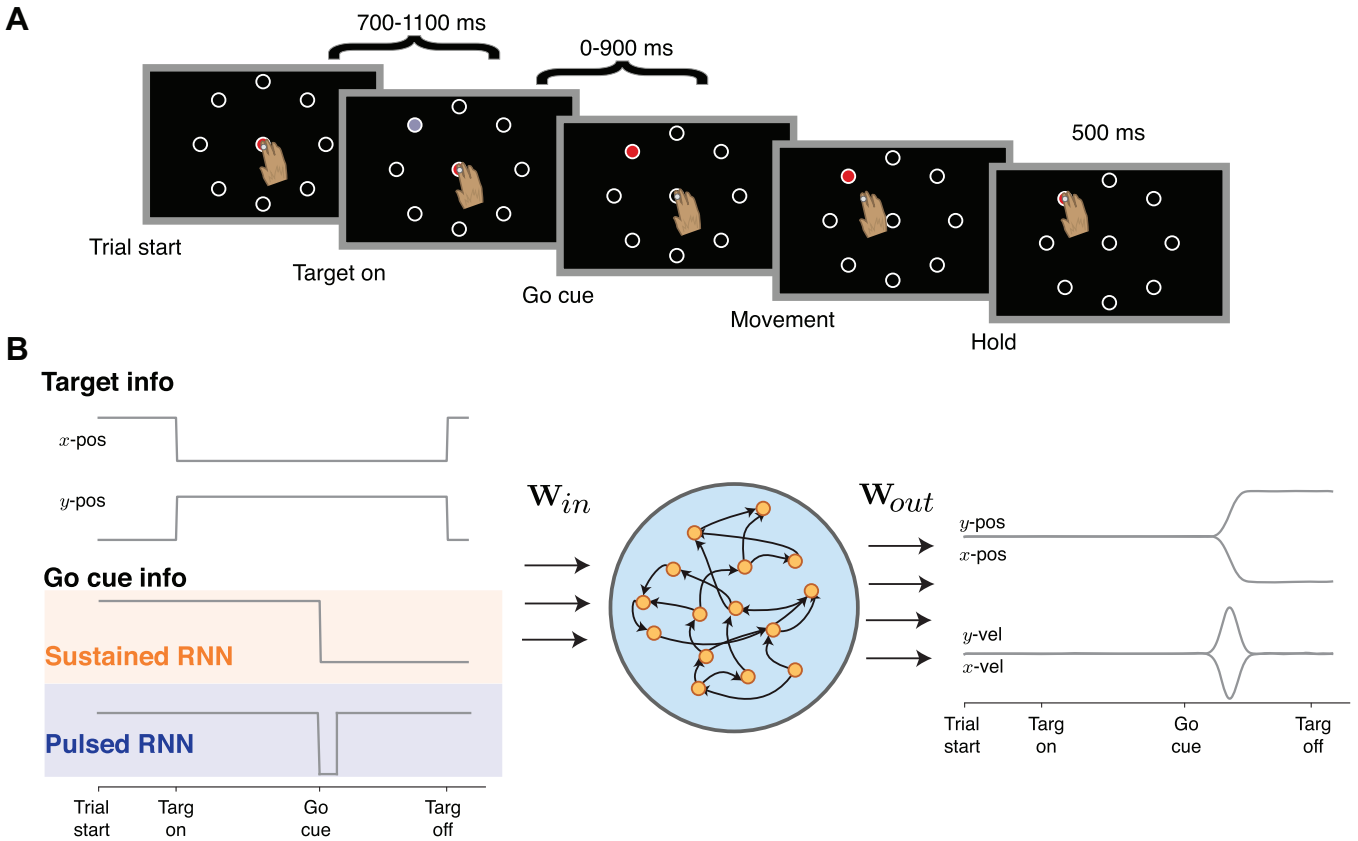

Fig. 2. Delayed reach task and recurrent neural network (RNN) training. *A*: schematic of the RNN task used by Ames and colleagues (2019). For RNN training, the hold time was allowed to be variable for anywhere from 500–1,500 ms so that the RNN did not learn to stay in a particular state for a specific time. *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 a desired position trajectory and velocity profile. The sustained RNN encodes the go cue as a signal to withhold movement (1) or to allow movement (0), whereas 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).

Table 1. *Parameters used for RNN training*

RNN Parameters	Values
No. of units	100
Time constant (τ)	50 ms
Discretization bin width	10 ms
L_2 regularizer for $\mathbf{W}_{in}$ (λ_{in})	1×10^{-3}
L_2 regularizer for $\mathbf{W}_{out}$ (λ_{out})	1×10^{-3}
L_2 regularizer for $\mathbf{W}_{rec}$ (λ_{rec})	1×10^{-3}
L_2 regularizer for $\mathbf{r}(t)$ (λ_r)	1.9×10^{-3}
λ_Ω regularizer	2
Activation function	tanh(·)
Initial learning rate (Adam)	1×10^{-4}
Maximum gradient norm	0.2

The λ_Ω regularizer is described further in the study by Pascanu et al. (2013) as well as Song et al. (2016). This regularization term approximately preserves the ratio of the backpropagated loss with respect to $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$, ameliorating the vanishing gradients problem. Its value was chosen through a heuristic hyperparameter search.

trials. After this target switch, the monkey was given a second delay period ranging from 0−900 ms before the go cue was delivered. We explicitly did not train on this task because we were interested in assessing how the RNN would generalize to it. We also incorporate trials to assess RNN generalization when the target is at an intermediate location [i.e., its radius is multiplied by a value $\alpha \in (0, 1)$].

Finally, we also quantified how closely artificial unit activity resembled activity from real data recorded while macaques performed a delayed reach task (Churchland et al. 2012). We note that this delayed reach task incorporated more conditions than the eight we trained on. First, for each artificial neuron, we report $r_{pm} = \mu_{prep}/\mu_{move}$, where μ_{prep} is the unit rate in the preparatory epoch across all eight target conditions, and μ_{move} is the mean rate in the movement epoch across all eight target conditions. This quantifies the ratio between mean preparatory and mean movement activity; a larger value indicates that the network has more preparatory activity. Second, we report $r_{prep} = \sigma_{prep}/\mu_{prep}$, where σ_{prep} is the standard deviation of the preparatory epoch rates across all conditions. This is a unitless quantity that is related to Fano factor (with standard deviation, instead of variance, in the numerator). It quantifies the variability in preparatory activity across conditions, normalized by the mean activity. Third, we report the entropy of the distribution of preferred directions over time in a preparatory epoch (200 ms preceding the go cue, H_{prep}) and the movement epoch (400 ms after the go cue, H_{move}). To calculate this entropy, for each artificial neuron, we first approximated its preferred direction in every 10-ms bin across the chosen epoch. This approximate preferred direction is the condition with the highest firing rate. We then determined the empirical distribution of these approximate preferred directions, enabling us to calculate the distribution's entropy. In real neural data, H_{prep} is relatively low entropy because preferred directions are largely static across time in the preparatory period. On the other hand, H_{move} is relatively high entropy because neuron firing rates during the move period are multiphasic, causing changes in preferred direction (Churchland and Shenoy 2007; Churchland et al. 2010; Michaels et al. 2016).

RESULTS

Before delving into design choices, we found that it was possible to train an RNN to capture key features of the neural activity in a delayed reach task, as reported in prior studies (Michaels et al. 2016; Sussillo et al. 2015). The hyperparameters for this network are listed in Table 1. Figure 3*A* shows PSTHs of artificial neurons for delayed reaches to eight different directions. These PSTHs plateau during the delay period (Churchland et al. 2010; Michaels et al. 2016; Sussillo et al. 2015; Tanji and Evarts 1976; Weinrich et al. 1984), have

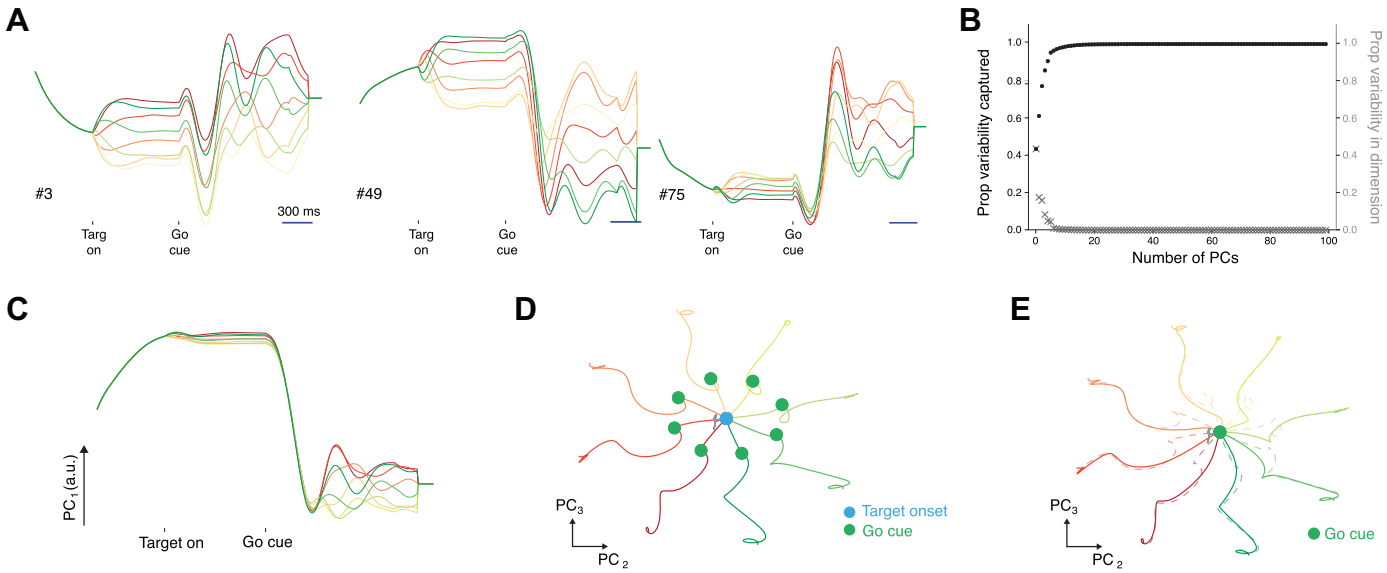

Fig. 3. Sample peristimulus time histograms (PSTHs) and population trajectories of the recurrent neural network (RNN). *A*: example PSTHs of artificial neurons in the RNN, where each color denotes one of the 8 reach conditions. The PSTHs achieve a stable firing rate in the preparatory period, followed 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. Five principal components (PCs) capture 90.8% of the PSTH variance, and 10 PCs capture 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 (2016). 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 before 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. We also show preparatory trajectories from *D*, denoted by dotted lines. This shows that the preparatory neural states are not obligatory, consistent with the findings of Ames and colleagues (2014). a.u., arbitrary units.

substantial heterogeneity and multiphasic activity during perimovement (Churchland and Shenoy 2007; Churchland et al. 2010; Sergio et al. 2005), and change preferred directions over time (Churchland and Shenoy 2007; Michaels et al. 2016), reflected by the fact that the condition with the highest firing rate is not the same across the entire trial. Figure 3, *B–E*, shows that the artificial neural population also captures qualitative observations from neurophysiologically recorded neural populations. We found that only 5 principal components (PCs) were required to capture more than 90% of the PSTH variance, as shown in Fig. 3*B*, demonstrating that the population is low dimensional (Ames et al. 2014; Cunningham and Yu 2014; Gallego et al. 2017; Gao and Ganguli 2015; Kao et al. 2015; Sadtler et al. 2014; Yu et al. 2009). We also found that PC_1, capturing 43.7% of the PSTH variance, strongly resembled a high variance condition-independent signal (Kaufman et al. 2016) (Fig. 3*C*), that artificial neural population activity had topographic organization in the PCs (Santhanam et al. 2009) (Fig. 3*D*), and that the neural population achieved a prepare-and-hold state attractor (Churchland et al. 2006) (Fig. 3*D*), but that this attractor was not obligatory (Ames et al. 2014) because the trajectories (for trials with no delay period) do not pass through the stable preparatory attractor (Fig. 3*E*).

Rate regularization and activation function affect preparatory activity. Prior studies using RNNs to model motor cortex use the hyperbolic tangent (tanh) activation function (Michaels et al. 2016; Sussillo et al. 2015) or a variant of it (Hennequin et al. 2014; Stroud et al. 2018). Recent studies have also used the rectified linear unit (relu) nonlinearity to model various decision-making tasks (Song et al. 2016, 2017). We note the relu nonlinearity has proliferated in several engineering applications, in part due to the faster training times and that the gradient of the relu is either zero or one, which is favorable for backpropagation (He et al. 2016; Krizhevsky et al. 2012; Szegedy et al. 2015). We found that the choice of activation function impacts population preparatory activity during the delay period. Preparatory activity captures a significant proportion of neural variance. Typically, preparatory activity plateaus to a stable level before movement onset (Churchland et al. 2010; Michaels et al. 2016; Sussillo et al. 2015; Tanji and Evarts 1976; Weinrich et al. 1984). In a dynamical systems framework, the population preparatory activity evolves to a subspace called the 'prepare-and-hold' state that is beneficial for the upcoming reach (Afshar et al. 2011; Ames et al. 2014; Churchland et al. 2006).

Given its prior use in RNN models of motor cortex, we first considered the hyperbolic tangent nonlinearity. Interestingly, we found that rate regularization (weighted by λ_r) was important for achieving population preparatory activity that was qualitatively consistent with neurophysiological data. When rate regularization was relatively small, we found that artificial neurons in the RNN had little population preparatory activity (Fig. 4*A*, leftmost panel). This can be observed by recognizing that population activity at the time of the go cue essentially overlapped with population activity at target onset. This solution is not unreasonable because the RNN's outputs remain zero during both the baseline and preparatory epochs. Although target information is available to the RNN in the preparatory period, it does not necessarily have to act (i.e., change its state) upon this information until the go cue is given. The RNN can therefore delay processing target information until the go cue is given and still successfully perform the task.

We found that, for the hyperbolic tangent nonlinearity, increasing rate regularization increased the amount of population preparatory activity in the network. This is shown for several values of λ_r in the remaining panels of Fig. 4*A* and summarized by Fig. 4*B*, which shows the ratio of lengths between the preparatory trajectory and the movement trajectory. The trajectory ratios are calculated in the high-dimensional artificial neuron activity space and not in the low-dimensional PCs. By observing the PSTHs of the activity at different levels of regularization (Fig. 4*E*), we find that rate regularization causes the rates to achieve *1*) smaller overall peak values and *2*) intermediate activations in the preparatory epoch. In this manner, rate regularization causes the tanh RNN to have stronger preparatory dynamics, effectively partitioning computation into two segments: preparatory dynamics (driving the activity to a attractor, denoted by the green dots in Fig. 4*A*) followed by movement dynamics (trajectory after the green dot). This is most apparent in the rightmost panel of Fig. 4*A*. This population activity is consistent with what is qualitatively observed in motor cortex (Afshar et al. 2011; Ames et al. 2014; Churchland et al. 2006).

We found that increasing rate regularization does not always result in more population preparatory activity. In fact, when we used the relu activation, we observed that the network finds a solution that has little preparatory activity on the population level, irrespective of λ_r (Fig. 4*C*; trajectory lengths summarized in Fig. 4*D*). We note that this was not because rate regularization was not 'active' due to other regularizations dominating the optimization cost; in fact, when we removed all regularization except rate regularization, the networks still had units with very little population preparatory activity across 4 orders of magnitude (from $\lambda_r = 10^{-3}$ to $\lambda_r = 1$). PSTHs of the tanh and relu RNNs are shown in Fig. 4, *E* and *F*. For relu RNNs in Fig. 4*F*, we observe that although the maximum rate may decrease as λ_r increases, the preparatory activity does not appear to increase in across-condition variability relative to movement activity. Furthermore, we point out that it was not the case that units with preparatory activity were absent in the relu network. Rather, Fig. 4*F* demonstrates that it was possible to find relu units that had preparatory activity. However, when quantifying the degree of preparatory activity in tanh units, relu units, and real data, we observed that relu units were skewed toward having less preparatory activity relative to movement activity (r_{pm}, Fig. 5*A*). Next, when quantifying the amount of preparatory variance across conditions (r_{prep}), we found that tanh units more closely matched the real data distribution, whereas relu units had a longer tailed distribution, reflecting low firing rate units (Fig. 5*B*). Finally, when analyzing the preferred directions of the units across time, we found that the tanh network had stable preferred directions in the preparatory period and changing preferred directions in the movement period, as observed in empirical data (Churchland and Shenoy 2007; Churchland et al. 2010). This was not the case for relu units, which had variable preferred direction in the preparatory epoch as a result of low firing rates. Both the tanh and relu units demonstrated high preferred direction entropy in the movement epoch, consistent with real data.

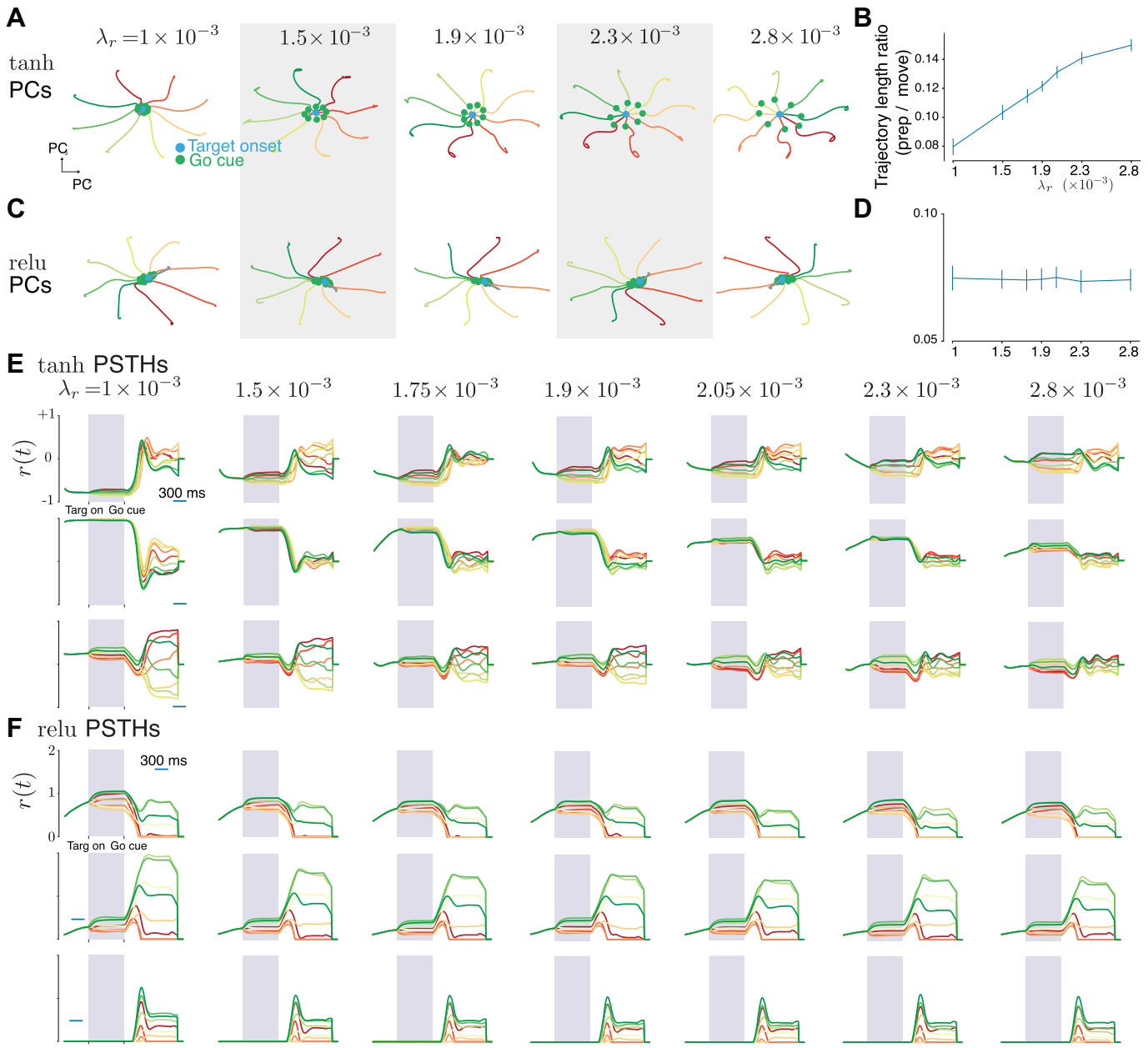

Fig. 4. Rate regularization increases population preparatory activity in hyperbolic tangent (tanh) but not rectified linear unit (relu) RNNs. *A*: population trajectories of tanh recurrent neural networks (RNNs) in principal components (PCs) 2 and 3 (analogous to Fig. 3*D*) for different values of λ_r, 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 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 population preparatory activity. *D*: same as *B* but for relu. *E*: peristimulus time histograms (PSTHs) for the same neuron in a tanh RNN across 7 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. *F*: PSTHs for the same neuron in an RNN across 8 different levels of regularization for a relu RNN. In general, even as rate regularization increases, the units have similar levels of preparatory and movement activity. We note that several relu units do have significant preparatory activity (e.g., third row).

Together, these results suggest that the relu RNNs do worse than the tanh RNNs at capturing key features in the neurophysiological activity at both the single-unit and population levels. Furthermore, we stress that when considering RNNs to model neurophysiological tasks, it is important not only to find single-unit examples that resemble physiological activity, but also to capture population-level features. Some relu RNN units had preparatory activity, but relu RNN units on the population level did not capture key features of preparatory activity.

Effects of other hyperparameters on RNN preparatory activity. Given the effect of rate regularization on the RNN's activity, we also considered how other regularizations, activation functions, and training constraints affected the net-

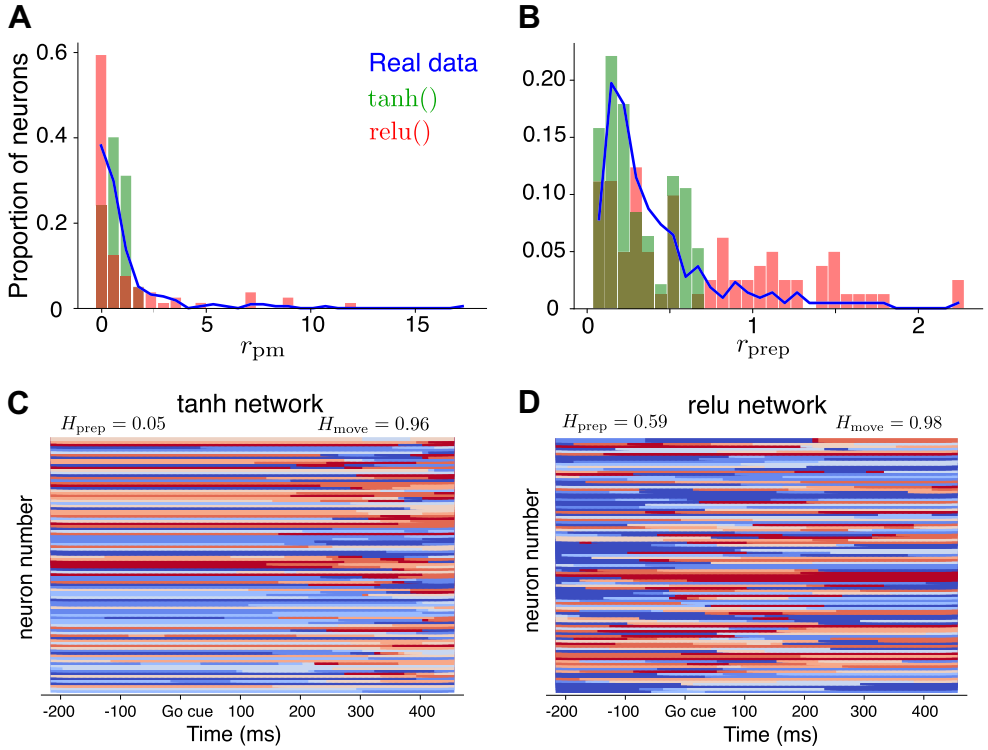

Fig. 5. Hyperbolic tangent (tanh) units have preparatory activity more consistent with data. *A*: r_{pm} histograms for a tanh recurrent neural network (RNN) (green) and a rectified linear unit (relu) RNN (red). We also calculated r_{pm} for real data shown in the blue line. The real data has a longer tailed distribution than the relu, which has a majority of cells without substantial preparatory activity. The tanh network, on the other hand, does achieve intermediate r_{pm} more consistent with the real data. *B*: r_{prep} histograms. The tanh RNN more closely resembles the distribution of the real data compared with the relu network. *C*: preferred directions over time for the tanh network. Each row corresponds to a neuron, and the *x*-axis represents time relative to the go cue (in ms). Each color corresponds to 1 of the 8 unique target reach conditions, and the color shown is the condition whose firing rate is highest. We observe large stability in the firing rates before the go cue (low entropy), but shortly after, we see preferred directions changing (high entropy). The reported entropies are averaged across all artificial units. This is consistent with prior results. *D*: same as *C* but for the relu network. Prior to the go cue, the preparatory periods are changing (in part because preparatory firing rates are very low for some units in relu networks), resulting in a high preparatory entropy.

work's preparatory activity. First, we varied the L_2 regularization strength for $\mathbf{W}_{\mathrm{in}}$ (given by λ_{in}), $\mathbf{W}_{\mathrm{rec}}$ (λ_{rec}), and $\mathbf{W}_{\mathrm{out}}$ (λ_{out}) in tanh RNNs. Over 6 orders of magnitude, we saw that modifying λ_{in}, λ_{rec}, and λ_{out} did not have a substantial effect in increasing the RNN's population preparatory activity, except when λ_{in} was on the order of 1 (Fig. 6*A*). Thus, it may be possible to use a large λ_{in} to achieve larger preparatory trajectories. It is worth noting that when we used $\lambda_{\mathrm{in}} > 1$, there was noticeable degradation in the RNN's performance in producing desired outputs. Overall, we found that the L_2 parameter regularization did not have as strong an effect on the network's preparatory activity as varying λ_r. Second, although studies have primarily used either the tanh (Chaisangmongkon et al. 2017; Mante et al. 2013; Sussillo et al. 2015), a variant of tanh (Hennequin et al. 2014; Michaels et al. 2016; Stroud et al. 2018; Sussillo et al. 2015), or the relu activation function (Song et al. 2016, 2017), we assessed whether other activation functions used in deep learning, in particular the leaky relu and the softplus (Yang et al. 2019) activation functions, would achieve

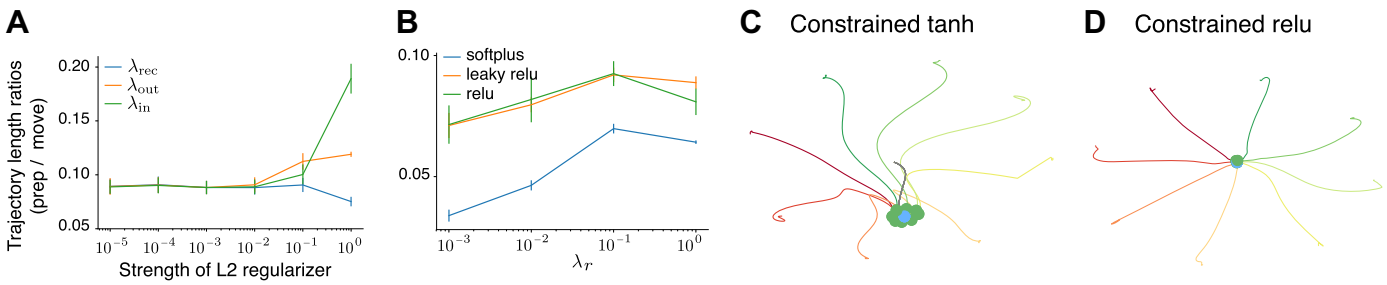

Fig. 6. Effects of selected hyperparameters on preparatory activity. *A*: for a hyperbolic tangent (tanh) recurrent neural network (RNN), varying the L_2 regularization strength for $\mathbf{W}_{\mathrm{rec}}$, $\mathbf{W}_{\mathrm{out}}$, and $\mathbf{W}_{\mathrm{in}}$ does not substantially increase population preparatory activity in the network, except for when $\mathbf{W}_{\mathrm{in}}$ is regularized very strongly ($\lambda_{\mathrm{in}} = 1$). *B*: varying the rectified linear unit (relu), leaky relu, and softplus activation function across different λ_r does not substantially increase population preparatory activity in the network. We note that at $\lambda_r = 1$ for the softplus activation, there was a noticeable degradation in the network's trained performance ($R^2 = 0.89$). *C*: trajectories for a tanh RNN trained with a constrained architecture having 80% (20%) excitatory (inhibitory) units and obeying Dale's law. The average preparatory-to-movement ratio is 0.074, significantly less than for the unconstrained networks. *D*: same as *C* but for a relu RNN. The average preparatory to movement ratio is 0.026, significantly less than for the unconstrained networks.

increasing preparatory activity with varying λ_r. As shown in Fig. 6*B*, we found that when sweeping λ_r over a large range, the leaky relu and the softplus networks did not have substantially more preparatory activity with increasing λ_r. This behavior resembles relu RNNs and contrasts tanh RNNs.

We also trained networks incorporating 80% excitatory units and 20% inhibitory units, following Dale's law (Song et al. 2016). We did not constrain the weights of $\mathbf{W}_{in}$ and $\mathbf{W}_{out}$ to be strictly positive. This is because there are several stages of processing between task inputs to the motor cortex, as well as between motor cortex and kinematics. For example, it need not be the case that all excitatory neurons in the RNN decrease their activity in response to the go cue. Rather, a study found that putative excitatory and inhibitory neurons in motor cortex both increase or decrease their firing rate following the go cue (Kaufman et al. 2013). We found that these constrained architectures had less preparatory activity in the population than the unconstrained architectures across both tanh and relu activity. The average ratio of trajectory lengths between preparatory and movement activity, for $\lambda_r = 1.9 \times 10^{-3}$, was 0.074 in tanh RNNs and 0.026 in relu RNNs, both of which are smaller than that of unconstrained networks (Fig. 6, *C* and *D*). We found that constrained tanh RNNs still had more preparatory activity than constrained relu RNNs. We also found that the dimensionality of the constrained RNN was smaller than the unconstrained, with 5 (10) dimensions capturing 96% (98.9%) of the PSTH variance in the tanh network. These results suggest that even though we are incorporating additional information about excitatory and inhibitory units and Dale's law into the RNN's architecture, the RNN does not necessarily increase its preparatory activity. For the rest of this work, we analyze RNNs with unconstrained architecture, using the parameters in Table 1. These RNNs utilize the tanh activation with rate regularization $\lambda_r = 1.9 \times 10^{-3}$ because it achieves excellent behavior and resembles neural population activity observed in motor cortex.

RNN dynamics in the sustained RNN during a delayed reach task. We next visualized the dynamics of the RNN (see MATERIALS AND METHODS) as displayed in Fig. 7. We also visualized the stable attractor regions of the dynamics by using the approach of (Sussillo and Barak 2013). We emphasize that these attractors were computed in the *N*-dimensional RNN state-space and subsequently visualized via projection into the low-dimensional subspace (see MATERIALS AND METHODS). We do not calculate attractors in a low-dimensional space; in general, these do not correspond to attractors of *Eq. 2*. We found that the RNN implemented a mechanism that can be interpreted as a composition of input-dependent dynamics to single stable attractor regions. These single stable attractor regions were the only attractor regions found by our optimization procedure (see MATERIALS AND METHODS). Key features of this mechanism were proposed by Ames and colleagues (2014) to describe neural population activity during a delayed reach task. In particular, Ames and colleagues (2014, 2019) proposed two principal dynamical systems: a 'preparatory' dynamical system implicated in planning a reach to a desired target, and a 'movement' dynamical system corresponding to the execution of the reach after the go cue is delivered. The preparatory dynamical system has a stable attractor corresponding to each prompted target. The neural state converges to this attractor, the prepare-and-hold state, during the delay period. This state is a favorable initial condition for the subsequent movement (Afshar et al. 2011; Churchland et al. 2006). When the go cue

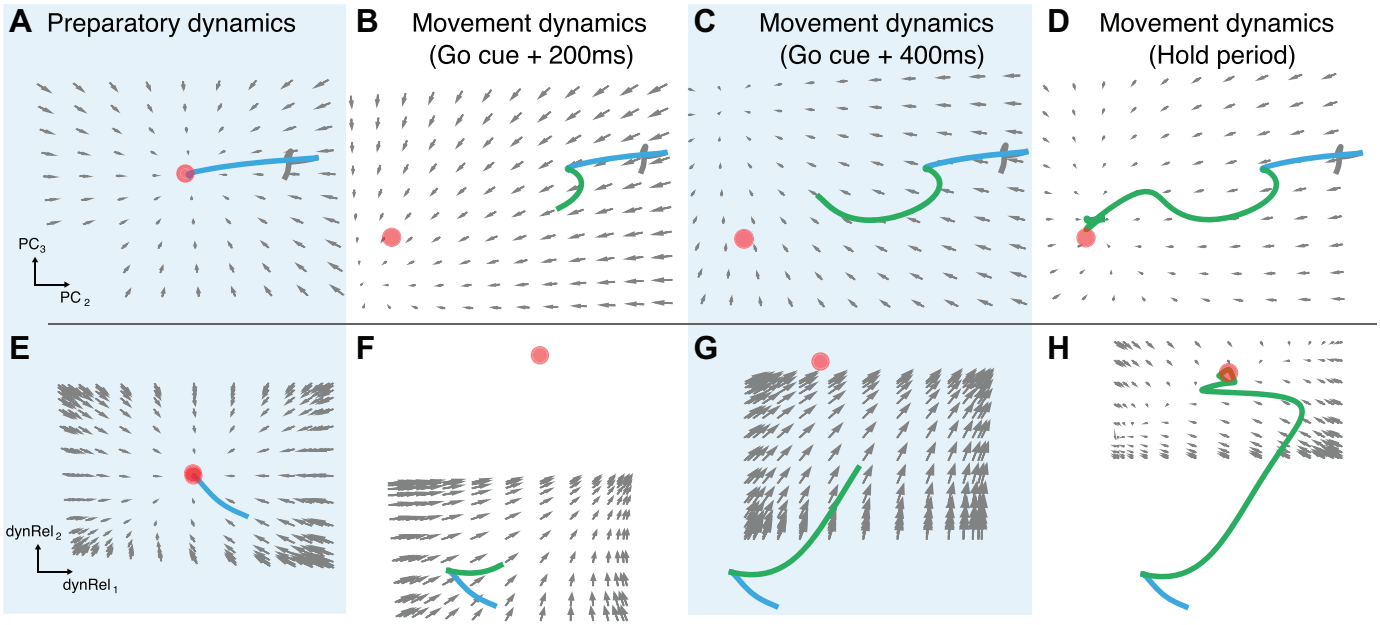

Fig. 7. Composition dynamical mechanism. *A*: delay period trajectory in principal components (PCs) 2 and 3. The gray portion of the trajectory corresponds to the baseline period of the task. The blue portion of the trajectory corresponds to the delay period of the task. The state converges along essentially linear dynamics to a stable attractor (shown as a red dot). *B*: 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 toward a single stable attractor region. There appear to be nonzero dynamics overlapping the attractor because the trajectory is not in the same plane as the attractor (in orthogonal dimensions). *C*: 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. *E–H*: same as *A–D* but using heuristic 2 (see METHODS AND MATERIALS) to determine a 'dynamics relevant' projection. This projection has dynamics that have consistent orientation in the movement epoch (*F* and *G*).

is given, the movement dynamical system is engaged, causing neural population trajectories associated with movement generation.

We visualized the RNN's dynamical equations to provide qualitative intuition for how the RNN uses nonlinear dynamics to perform the task. We found that during the delay period, the RNN implemented dynamics analogous to the preparatory dynamical system. Upon target presentation, the trajectories were driven to a single stable attractor region, found by our full-dimensional RNN attractor optimization, as in Fig. 7*A*. This stable attractor location was target dependent. The RNN achieved different preparatory attractors and dynamics for a given target because the network's dynamics at a given time, $\dot{\mathbf{r}}(t)$, are modified by an input-dependent additive factor, $\mathbf{W}_{\text{in}}\mathbf{u}(t)$. This input modifies the overall RNN dynamics, $\dot{\mathbf{r}}_t$, and therefore the trajectory. This enables the network to, before the go cue, instantiate different stable attractor regions and preparatory dynamics for each prompted target.

When the go cue was delivered, the changing input considerably modified the dynamics. We found that these dynamics drove the RNN state to a different stable attractor region, analogous to the movement dynamical system. These dynamics caused the RNN's rate trajectories to move at relatively high speeds, as shown in Fig. 7, *B* and *C*. In this manner, the mechanism is not integration along slow points on a line attractor (Mante et al. 2013), but rather abrupt transitions from stable attractor to stable attractor. A video of these dynamics is shown in Supplemental Movie S1. We also have included, in Fig. 7, *E–H*, the same projections visualized using heuristic 2 in the MATERIALS AND METHODS (see *Visualizing RNN dynamics*). In particular, the projections in Fig. 7, *F* and *G*, which correspond to different times during the movement epoch, are more similarly oriented than in the PC visualization of Fig. 7, *B* and *C*. These types of strong dynamics that drive the network state to stable attractors, illustrated in Fig. 1*C*, have been observed in another study (Chaisangmongkon et al. 2017). An increase in the speed of neural trajectories following the go cue is consistent with experimental observations from dorsal premotor cortex (PMd) and M1 (Afshar et al. 2011) (their Fig. 3*C*). Note that when target presentation is simultaneous with the go cue so that there is no delay period, the movement epoch dynamics are immediately engaged and trajectories are driven to its single stable attractor region. Because the preparatory epoch dynamics have not been engaged for enough time, the trajectories do not achieve the preparatory attractor, a phenomena also observed in neurophysiological data (Ames et al. 2014).

RNNs qualitatively capturing neurophysiological motifs may utilize different dynamical mechanisms. We next wondered if task design considerations could produce RNNs that, while looking qualitatively similar to neurophysiological data, utilize distinct dynamical mechanisms. To assess distinct dynamical mechanisms, we computed the stable attractor regions of the full-dimensional RNN and subsequently visualized the RNN dynamics to gain qualitative insight. We trained the pulsed RNN described earlier to perform a delayed reach task using the same hyperparameters as the sustained RNN. Our goal was to assess whether the sustained and pulsed RNNs utilize the same dynamical mechanisms. In the training set, the pulsed go cue was delivered for 150 ms. This pulse may also be interpreted as reflecting that the state of the task has changed (Remington et al. 2018) so that the animal may now make a reach, analogous to a signal that triggers movement (Erlhagen and Schöner 2002; Kaufman et al. 2016). We are not suggesting that this pulse length would be reasonable for experiments;

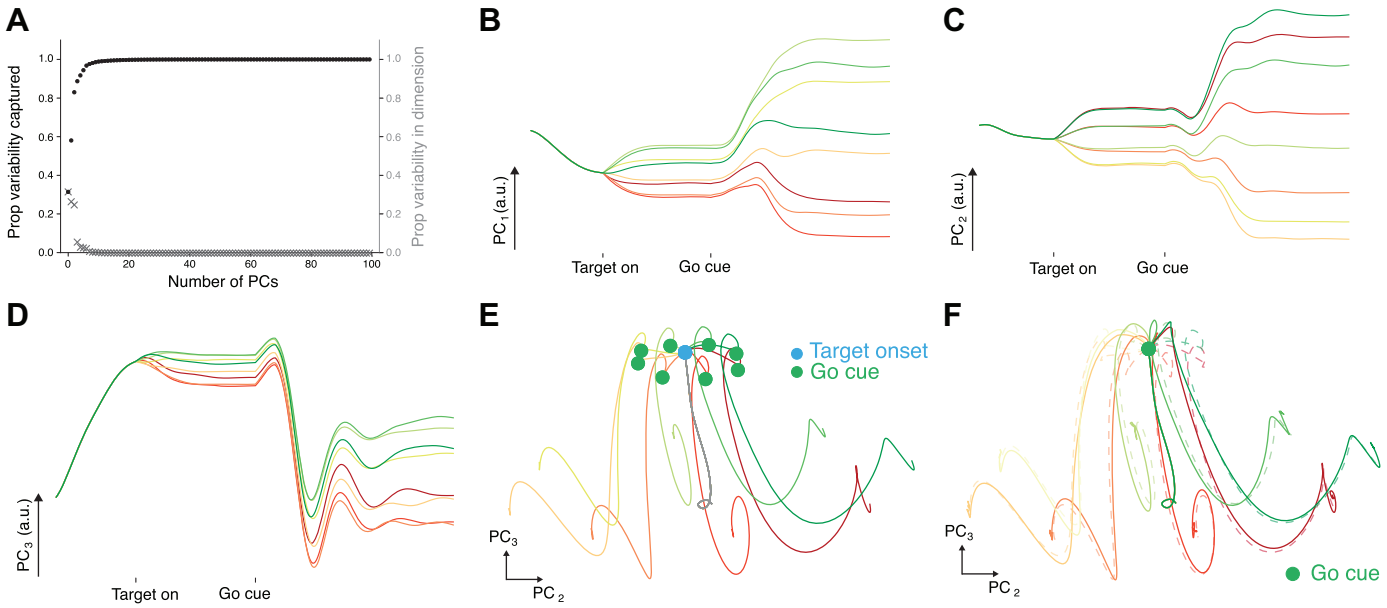

Fig. 8. Principal components analysis for recurrent neural network (RNN) trained to perform a delayed reach task with a pulsed go cue. *A*: variance captured by dimension. The first 5 principal components (PCs) capture 91.8% of the peristimulus time histogram (PSTH) variance and the first 10 PCs capture 98.6% of the PSTH variance. *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, and the solid traces are trajectories without a delay period.

although we chose to use 150 ms, this pulse length can be varied. The RNN was capable of performing the pulsed go-cue task at the same level as the single attractor RNN (training terminated when $R^2 > 0.997$ on validation data, example output trajectory shown in dark blue in Fig. 9*A*). Its recurrent computation was similarly low dimensional, with 5 PCs capturing 91.8% of the PSTH variance (Fig. 8*A*). This RNN also bore hallmarks of neurophysiological responses, including neural activity being organized topographically (Fig. 8*E*), the trajectories achieving a 'prepare-and-hold' state in the delay period (Fig. 8*E*), and that these states were not obligatory (Fig. 8*F*). We do note that condition-independent variance, although present, appeared to be smaller in this network, with a large proportion appearing in PC 3 (Fig. 8, *B–D*).

In this task design, the input during the delay period is the same as the input during the movement period post go pulse. Because the RNN's parameters are fixed and the inputs are equal, the attractors in the delay period (pre go pulse) and movement period (post go pulse) are the same. Therefore, this RNN cannot use a composition of dynamics to single stable attractor regions. Doing so would imply that the delay and movement periods must converge to the same stable attractor, and hence the delay and movement periods would converge to the same output under this mechanism. Such an RNN would be unable to adequately perform this task. We note that although we have chosen a pulsed go cue, a similar conclusion holds for RNNs that use transient cues, such as the networks trained by Hennequin and colleagues (2014) with a movement instruction cue. Analogously, this network produced two different output transients (prepreparatory and postmovement) for the same input.

Although the trajectories in condition-relevant dimensions demonstrate qualitatively similar trajectories to neurophysiological data, how does the RNN dynamically achieve this, if not by the mechanism used by the sustained RNN? To answer this question, and recognizing that the RNN must be capable of achieving two steady-state outputs, we pulsed the go cue to determine what duration of go cue was required for the pulsed RNN to settle to the correct final kinematics as opposed to returning to the preparatory state kinematics (i.e., zero posi-

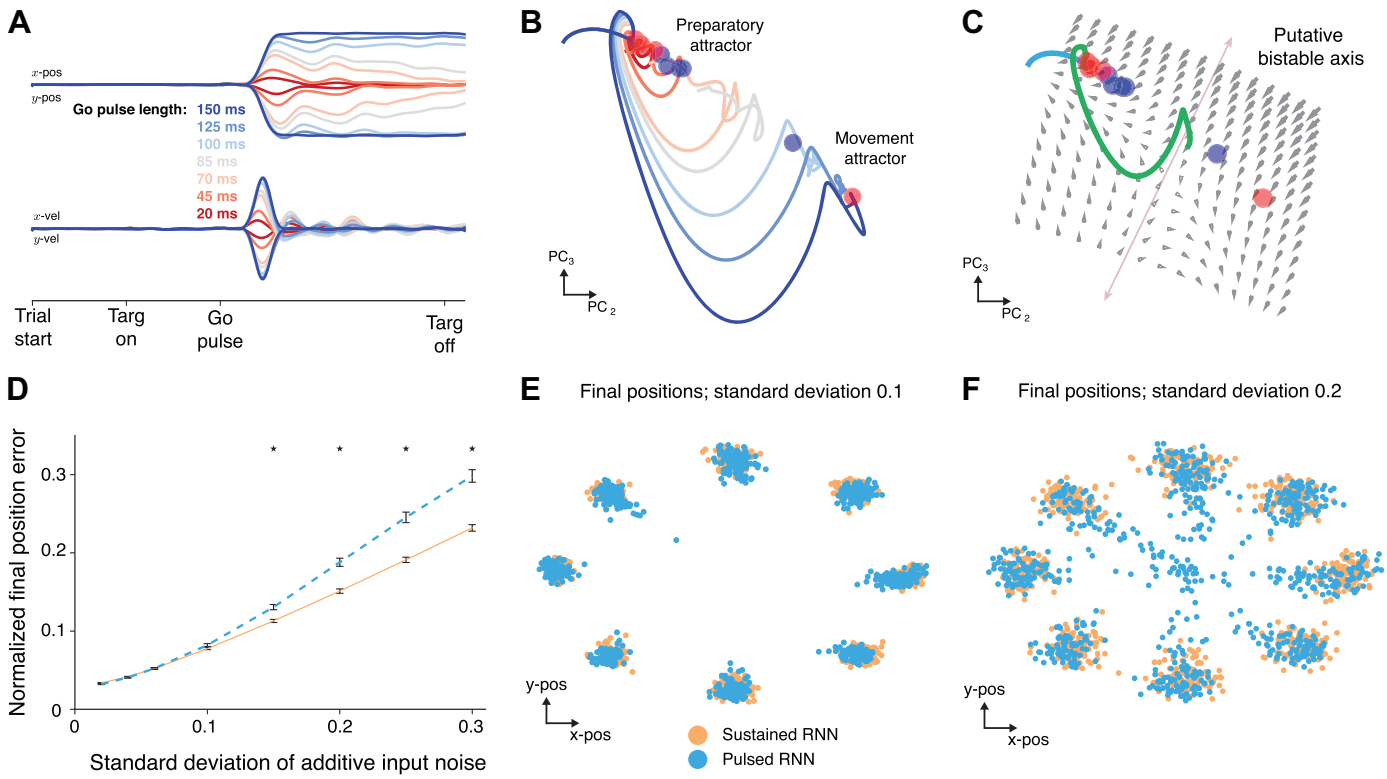

Fig. 9. A recurrent neural network (RNN) trained to perform a delayed reach task with a transient go cue. *A*: example output for an RNN that was trained on a pulsed go cue task to make a reach to the target at 315°. The output is shown for pulsing the go cue at different lengths, denoted by different 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 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*: dynamics at the point of bistability. A putative bistable axis, from inspecting dynamics, is illustrated in light purple. Left of the axis, dynamics drive the trajectory to decay back to the preparatory attractor. Right of the axis, dynamics 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, and the solid line represents the performance of the sustained RNN. As input noise increases, the pulsed RNN has worse final position performance. *Significant differences in the mean (bootstrap, 1,001 shuffles, $P < 0.01$). *E*: final positions to the 8 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 decays back to the center target, which is the kinematic output corresponding to the preparatory attractor. The hold time was increased to 2,000 ms to show this slow decay. Trials that 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. PC, principal component.

tions and velocities). We delivered different input pulses, as shown in Fig. 9*A*, and found that when the pulsed go cue was delivered for less than 85 ms, the pulsed RNN produced transient kinematics that decayed back to zero. However, when the go cue was delivered for greater than 85 ms, the pulsed RNN produced the correct final kinematic output corresponding to the prompted reach. We note that this pulse length is model dependent; training with a different pulse length would also change the maximum pulse length for which the model decays back to zero.

The pulsed RNN sets up two stable attractor regions. Our full-dimensional RNN attractor optimization did not identify any other attractors, suggesting that the pulsed RNN implements a bistable dynamical system. We visualized the trajectories and dynamics, as shown in Fig. 9, *B* and *C*. One stable state is associated with the preparatory period, analogous to the prior RNN, where the RNN rates converge to during the delay period. The second stable state is the region associated with making a movement to the prompted target. In Fig. 9*C*, we were able to visualize bistable dynamics associated with the task. These bistable dynamics could be viewed in the projections of PCs 2 and 3. For intuition, we illustrate a putative bistable axis, around which dynamics diverge in opposing directions. The go-cue pulse drives the trajectory to this region of bistability, and depending on the trajectory's location, it will either settle back to the preparatory state (with zero kinematic output, i.e., left of the illustrated bistable axis) or settle to the state associated with the correct kinematic output (i.e., right of the illustrated bistable axis).

These results demonstrate that for trajectories having qualitative similarities to neurophysiologically observed data, different dynamics may be at play. We found that the employed mechanism differed substantially from task input design (i.e., strongly driving trajectories to single attractors vs. implementing a region of bistability). To evaluate these mechanisms, we asked which mechanism was more robust to input noise, as could occur from suboptimal processing of the task inputs. We added independent zero-mean Gaussian noise to the inputs and assessed the RNN's performance as a function of the standard deviation of the Gaussian noise.

We found that increasing the input noise affects the network in at least two distinct ways. First, for both RNNs, because the stable attractor region is input dependent, noisier inputs cause the stable attractor region to be variable, resulting in greater neural trajectory end-point variability and hence kinematic end-point variability. We observed this effect, as end point deviation increased with the standard deviation of Gaussian noise, shown in Fig. 9*D*. Interestingly, however, we did not observe a significant difference in performance between the sustained and pulsed RNNs when the standard deviation was less than or equal to $\sigma = 0.1$, which is ~10% of the input signal ($P = 0.69$, bootstrap with 1,001 shuffles). These variable end-point positions, for $\sigma = 0.1$, are shown in Fig. 9*E*. The effect of input noise varying attractor location was similar in both networks. Second, we found that for the pulsed RNN, input noise may result in final positions closer to zero. For sufficient noise, the RNN state may not cross the bistable axis, resulting in eventual relaxation to the preparatory attractor. As shown in Fig. 9*D*, when the standard deviation increased beyond $\sigma = 0.15$, the pulsed RNN had worse final end-point performance than the single attractor RNN ($P < 0.01$, bootstrap with 1,001 shuffles). By visualizing kinematic end points when $\sigma > 0.15$, we found that the RNN outputs are closer to the (0, 0) output center position, consistent with relaxation to the preparatory attractor. This demonstrates that in the presence of noise, RNN task performance will similarly degrade until the point where noise causes the pulsed RNN's state to converge to the incorrect attractor. This suggests that for the purposes of performing a delayed reach task, the sustained RNN is more robust under input noise.

RNN generalization to new tasks. Finally, we assessed the extent to which a qualitative understanding of the RNN's dynamics could inform task generalization. We believe this is an important line of questioning for future RNN studies. In particular, by constraining what tasks the RNN is trained on, it is possible to comment on what dynamical mechanisms are sufficient to carry out certain tasks. As an example, can an RNN using the composition of preparatory and movement dynamics generalize to perform a target switch task? The target switch task is diagrammed in Fig. 10*A*. We hypothesized this should be possible; indeed, Ames and colleagues (2014) used a qualitatively similar dynamical mechanism to describe how motor cortex performs a target switch task. By training an RNN to only perform a delayed reach task, we can assess whether this dynamical mechanism is sufficient to enable the network to perform related tasks it was not trained on. We considered three variants of a target switch task, where the target switches at different times: *1*) before the go cue, *2*) simultaneous with the go cue, and *3*) after the go cue. The first two were considered in Ames et al. (2014), and the third was considered in Ames et al. (2019).

Consider first the sustained RNN. When the target switch is delivered before the go cue, the preparatory dynamical system is changed from that associated with the before-switch target to that of the after-switch target. As a result, the preparatory stable attractor changes when the target is switched, and the RNN's rates will converge to the single stable attractor associated with the switched target. When the go cue is then delivered, the RNN will execute the reach as it did in a delayed reach task. We found this was the case, as illustrated in Fig. 10, *B–E*, mimicking the experimental results presented in Ames et al. (2014). When the go cue is given simultaneously with the target switch, this is analogous to performing a delayed reach from a suboptimal initial condition. The network will achieve the correct end behavior because when the go cue is delivered, it must settle to the single stable attractor region of the movement dynamical system, as shown in Fig. 10, *F–I*. However, there is the potential for initial aberrant kinematic activity from initiating the movement dynamical system from the attractor of an incorrect target plan. We found that this was not the case, and indeed the RNN was able to carry out the target switch task with no additional delay period, illustrated by representative examples in Fig. 11*A*. Thus, the mechanism used by the RNN to perform the delayed reach task generalizes to perform two variants of the target switch task, even though it was not trained on it. Analogous arguments apply to the pulsed RNN. We found that, like the sustained RNN, the pulsed RNN also generalized to the target switch task when the switch was delivered either at the same time or preceding the go cue, as shown in Fig. 11*A*.

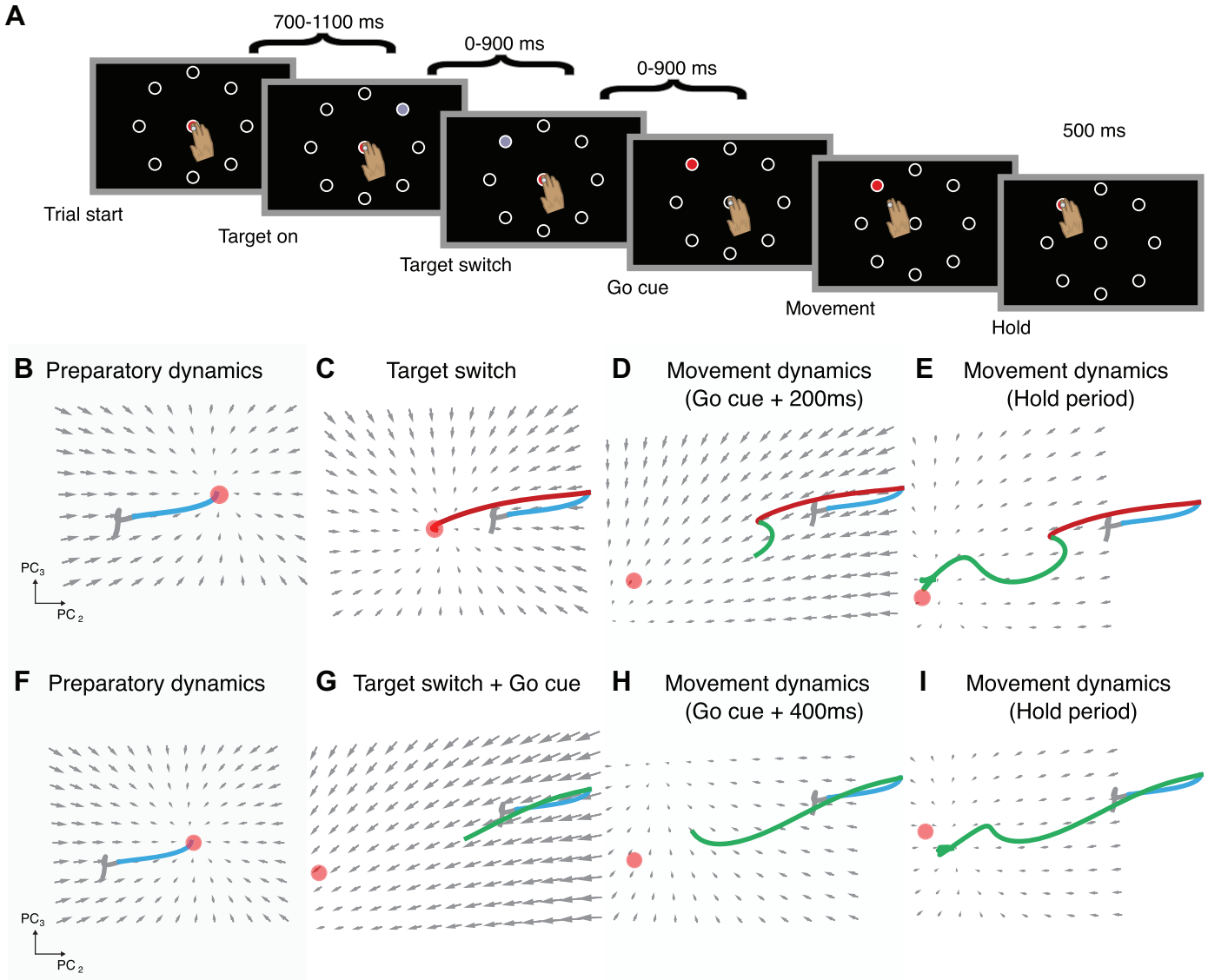

Fig. 10. Recurrent neural network (RNN) dynamics during a target switch task. *A*: schematic of the target switch task when the go cue is delivered either with a delay or contemporaneously with the go cue. *B–E*: RNN dynamics during a target switch task, with an additional delay period to reprepare. *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* and *E*: dynamics following the go cue, resembling the trajectory seen in Fig. 7, *C* and *D*. *F–I*: same as *B–E* but when the go cue is given contemporaneously with the target switch.

We also considered a task when the target is switched after the go cue. This task comprises an online corrective component, where feedback of the arm's kinematics play an important role in updating motor commands to reach to a new target. Although we did not account for this corrective component, we nevertheless found that the open-loop RNN could reasonably perform the task. This is shown in Fig. 11, *B* and *C*, where for different lengths of time after the go cue, a target switch is delivered in the middle of the trial. The RNN converges to the correct final output, as expected, because it will converge to the stable attractor location corresponding to the switched target input. We found that even though there was not a corrective feedback component and the target was abruptly changed, the RNN made a smooth and reasonable trajectory between targets.

For target switches after go cue, we found that the pulsed RNN had poorer generalization in the presence of input and recurrent noise. Recurrent noise is directly added to the network's state. We incorporated input noise and recurrent noise into RNNs as they performed a task where the target switched 200 ms after the go cue. We found that, in general, the pulsed RNN had poorer robustness to both input noise and recurrent noise (Fig. 12, *A* and *B*) across varying levels of noise. In particular, we found that the pulsed RNN especially performed worse when the target switch was maximal at 180° (purple lines in Fig. 12, *A* and *B*). The sustained RNN adequately performed this task, being able to make diagonal corrections, as shown in Fig. 12*C*. However, we found that the pulsed RNN was not able to consistently perform this task (example trajectories in Fig. 12*D*). One reason for poorer performance is that on several occasions, the output trajectories began to correct in the right direction but relaxed back to the zero position. This is consistent with the RNN state not crossing the bistable axis and relaxing back to the preparatory state attractor. These results

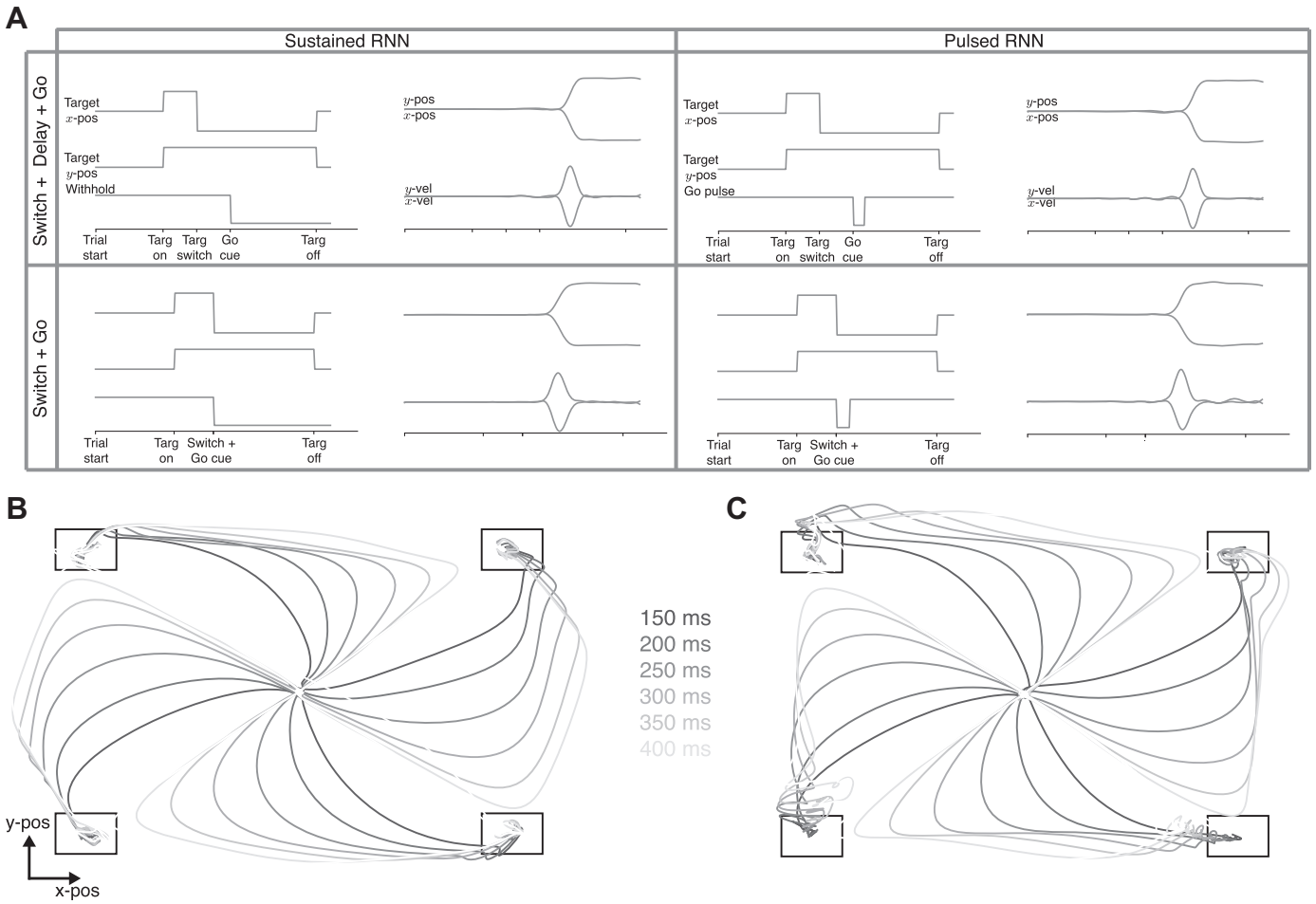

Fig. 11. Example behavior for the recurrent neural networks (RNNs) generalizing to perform the target switch task. *A*: both RNNs discussed generalize to the target switch task when the target switch occurred either before or at the same time as the go cue. Both RNNs had a coefficient of determination $R^2 > 0.99$ 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 4 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. Different shades correspond to output trajectories when the target was switched some amount of time following the go cue (legend between *A* and *B*). *C*: same as *B*: but for the pulsed RNN trained on the pulsed go-cue delayed reach task.

suggest that although both the sustained and pulsed RNNs are capable of using their dynamical mechanisms to generalize to target switch tasks, the sustained RNN has more robust generalization in the presence of noise.

Lastly, we also consider generalization to generate trajectories to targets that were not presented in the training set. To do so, we scaled the radius of the targets by a scale factor, $\alpha \in (0, 1)$, so that presented targets were closer than those in the training set. Even when trained on targets of only one radius, we found that both the sustained and pulsed RNNs were able to scale down their output position and velocities, although they tended to overestimate the desired output, as shown in Fig. 13, *A* and *B*. These results are consistent with the inputs modifying the location of the target-dependent stable attractor, enabling the network state to achieve a scaled down final position and velocity profile. Interestingly, we also noticed that the sustained RNN had superior generalization to the pulsed RNN when the target radius was decreased, as shown in Fig. 13*C*. In particular, when the target radius decreased significantly, we observed increased vacillation in the pulsed RNN outputs. This observation is consistent with the two stable attractors of the pulsed RNN becoming relatively close, so that the network state moves around two adjacent regions of slow dynamics. Together with the target switch task, these simulations demonstrate that the sustained RNN generalizes more effectively and is more robust to noise than the pulsed RNN.

DISCUSSION

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

First, we found that even when some RNN units resemble single neuron motifs, they may not faithfully reproduce population level motifs in the data. This is clear in the choice of

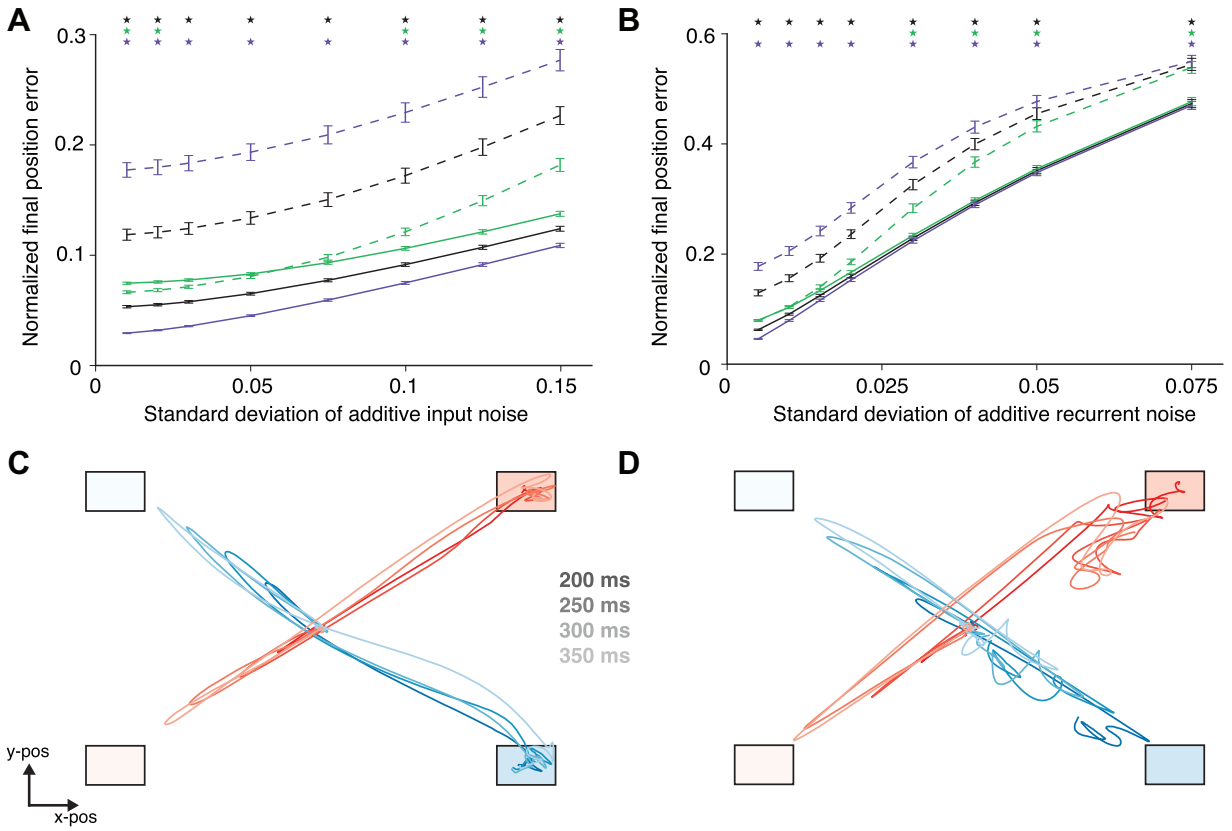

Fig. 12. *A*: normalized error in the final position as a function of the standard deviation of Gaussian noise added to the input. The final position was measured at 1,200 ms after go-cue onset, when the trials were terminated. The position errors are normalized so that the target position has a distance of 1. 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 dotted lines correspond to the performance of the pulsed recurrent neural network (RNN). Black lines denote the error across all switch conditions. Purple lines denote the error for switch conditions where the switched target was 180° away from the preswitch target, and green lines correspond to the error for switch conditions where the switched target was +90° away from the preswitch target. *Significant difference in the means at the level $P < 0.01$ (bootstrap, 1,001 shuffles). Error bars are standard error of the mean. In general, the pulsed RNN has poorer robustness under additive input noise. *B*: 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 2 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 an incorrect final position.

activation function to train RNNs. In RNNs trained with the relu activation, it was possible to find artificial neurons with preparatory activity, but these RNNs did not capture preparatory activity across the population. Furthermore, finer quantitative metrics on the single neuron level could identify mismatches across the population in comparison with real data (Fig. 5). For relu RNNs, preparatory trajectories were relatively short compared with movement trajectories (Fig. 4*C*), which is inconsistent with empirical results. Furthermore, even for the tanh activation function, we found that it was important to regularize the network rates to capture preparatory variability in the population. This would not have been straightforward if only qualitatively comparing single-neuron PSTHs.

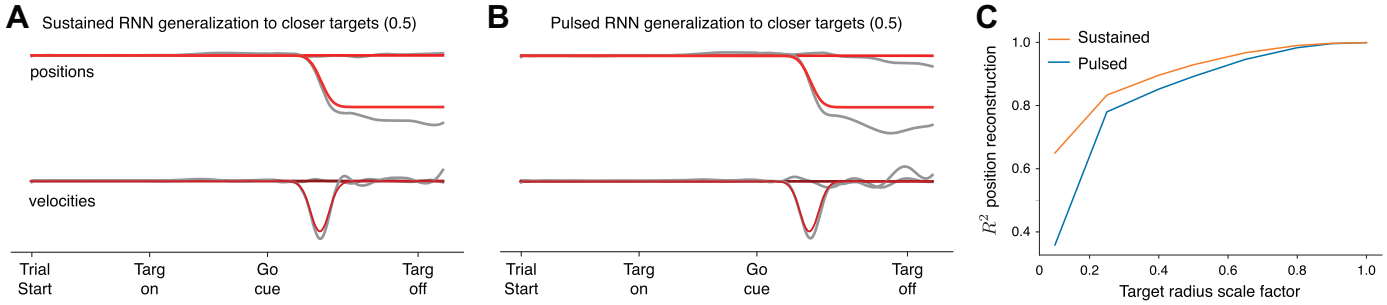

Fig. 13. Generalization to closer targets. *A*: example trial for the sustained recurrent neural network (RNN), outputting a desired trajectory to a target at half the radius of the training set targets. The desired output is in red, and the RNN's output is shown in gray. The RNN achieves a larger final position and peak velocity than desired. *B*: same as *A* but for the pulsed RNN. Its generalization is noticeably worse than the sustained RNN. *C*: as the target radius decreases, the sustained RNN considerably outperforms the pulsed RNN in reconstructing the desired output.

Second, we saw that two distinct RNNs could both capture key hallmarks of neurophysiological activity, but do so with fundamentally different mechanisms. In this manner, even if an RNN reproduces both single neuron and population level motifs, careful consideration should be given as to how the RNN dynamically performs the task. Our results showed that by varying how the task inputs are designed, the RNN can use distinct mechanisms with important consequences on generalization in the presence of noise. Our results also demonstrate that in addition to regularizations (e.g., Sussillo et al. 2015), architectures (e.g., Song et al. 2016), and training rules (e.g., Miconi 2017; Song et al. 2017), task input design can have an important effect on how the RNN's computations are performed. It may be possible for future experiments to be designed in such a way as to provide evidence in favor of a particular mechanism by assessing robustness in the presence of noisy inputs and generalization.

As task design can affect the dynamical mechanisms employed by the RNN, so too may other hyperparameters and training paradigms. It will be appropriate to consider how RNN architectures and parameters affect dynamical mechanisms, their robustness to noise, and generalization. For example, Song and colleagues (2016) demonstrated that it is possible to design RNNs that obey biological constraints such as Dale's law. They demonstrated that in these networks, one could find single artificial neurons consistent with empirical recordings. We found that implementing an 80:20% population of excitatory-to-inhibitory units following Dale's law caused the RNN to have less preparatory data, suggesting that these architectural constraints may not always cause the network to more strongly resemble recorded neurophysiological activity. Nevertheless, these constraints are important to consider thoughtfully in RNN design. For example, a different study we performed also identified that incorporating multiple area computation into RNN architectures is important to faithfully reproduce neurophysiological activity in PMd during a perceptual decision-making task (Kleinman et al. 2019).

It will also be important to consider how different training rules affect dynamical mechanisms. For example, Miconi (2017) demonstrated that networks can be trained with biological learning rules. These learning rules, which reproduce neurophysiological features of the data, may affect the network dynamics. Assessing the extent to which important features of the employed dynamical mechanisms change through introducing biological constraints and learning may play an important role in proposing mechanisms for cortical computation and making concrete predictions for future experiments.

Interrogating an RNN's dynamics also has consequences for what type of dynamics may be sufficient to carry out a class of tasks. In our work, we found that an RNN trained to only perform a delayed reach task was capable of generalizing to target switch tasks, even though it was not trained on these tasks. This shows that the mechanism employed by the RNN to perform a delayed reach task endows the network with the capability of performing the target switch task. An interesting line of future work may assess how the RNN's dynamics change as it is trained to perform a wider assortment of tasks (Yang et al. 2019). This may describe how many and what classes of tasks are necessary to provide an RNN with the capability of performing a different set of tasks. Furthermore, insofar as the capability to perform a variety of tasks changes the dynamical mechanisms of the network, this may help to narrow the set of plausible mechanisms used to perform a given task. For example, if we trained the networks in this work to perform motor tasks with perturbations to the arm (e.g., Nashed et al. 2012; Omrani et al. 2014), does the network cease to employ a bistable mechanism? Another line of inquiry is to visualize the dynamics of RNNs that fail to generalize and determine what deficiencies result in poor generalization.

The trained RNN outputs were desired feedforward kinematics. Although prior work modeling a delayed reach task has also used these outputs (Michaels et al. 2016), studies have also modeled electromyographic outputs (Hennequin et al. 2014; Stroud et al. 2018; Sussillo et al. 2015). The output of the RNN may significantly impact the computations it performs. Furthermore, RNNs to date do not model the reaction time of movements. Rather, RNNs typically output kinematics or electromyographics associated with a corresponding target at a fixed reaction time (Hennequin et al. 2014; Michaels et al. 2016; Stroud et al. 2018; Sussillo et al. 2015). Even with these output limitations, we nevertheless observed resemblance between the RNN's artificial activity and neural population responses from motor cortex. Although this work assessed how RNN inputs affect its dynamics, future work should assess how RNN outputs may impact RNN dynamical mechanisms.

ACKNOWLEDGMENTS

We thank Chandramouli Chandrasekaran and Krishna Shenoy for helpful discussions. We also thank Mark Churchland, Matt Kaufman, and Krishna Shenoy for permission to use the data set associated with the jPCA toolbox (Churchland et al. 2012). To train RNNs, we used pycog (https://github.com/xjwanglab/pycog/) with modifications for task design, architecture, and optimization.

GRANTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. This work was also supported by a UCLA Computational Medicine Amazon Web Services award.

DISCLOSURES

No conflicts of interest, financial or otherwise, are declared by the author.

AUTHOR CONTRIBUTIONS

J.C.K. conceived and designed research; performed experiments; analyzed data; interpreted results of experiments; prepared figures; drafted manuscript; edited and revised manuscript; approved final version of manuscript.

REFERENCES

Afshar A, Santhanam G, Yu BM, Ryu SI, Sahani M, Shenoy KV. Single-trial neural correlates of arm movement preparation. *Neuron* 71: 555–564, 2011. doi:10.1016/j.neuron.2011.05.047.

Ames KC, Ryu SI, Shenoy KV. Neural dynamics of reaching following incorrect or absent motor preparation. *Neuron* 81: 438–451, 2014. doi:10.1016/j.neuron.2013.11.003.

Ames KC, Ryu SI, Shenoy KV. Simultaneous motor preparation and execution in a last-moment reach correction task. *Nat Commun* 10: 2718, 2019. doi:10.1038/s41467-019-10772-2.

Boyd S, Vandenberghe L. *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

Chaisangmongkon W, Swaminathan SK, Freedman DJ, Wang XJ. Computing by robust transience: how the fronto-parietal network performs

sequential, category-based decisions. *Neuron* 93: 1504–1517.e4, 2017. doi:10.1016/j.neuron.2017.03.002.

Chandrasekaran C. Computational principles and models of multisensory integration. *Curr Opin Neurobiol* 43: 25–34, 2017. doi:10.1016/j.conb.2016.11.002.

Churchland MM, Cunningham JP, Kaufman MT, Foster JD, Nuyujukian P, Ryu SI, Shenoy KV. Neural population dynamics during reaching. *Nature* 487: 51–56, 2012. doi:10.1038/nature11129.

Churchland MM, Cunningham JP, Kaufman MT, Ryu SI, Shenoy KV. Cortical preparatory activity: representation of movement or first cog in a dynamical machine? *Neuron* 68: 387–400, 2010. doi:10.1016/j.neuron.2010.09.015.

Churchland MM, Shenoy KV. Temporal complexity and heterogeneity of single-neuron activity in premotor and motor cortex. *J Neurophysiol* 97: 4235–4257, 2007. doi:10.1152/jn.00095.2007.

Churchland MM, Yu BM, Ryu SI, Santhanam G, Shenoy KV. Neural variability in premotor cortex provides a signature of motor preparation. *J Neurosci* 26: 3697–3712, 2006. doi:10.1523/JNEUROSCI.3762-05.2006.

Cunningham JP, Yu BM. Dimensionality reduction for large-scale neural recordings. *Nat Neurosci* 17: 1500–1509, 2014. doi:10.1038/nn.3776.

Erlhagen W, Schöner G. Dynamic field theory of movement preparation. *Psychol Rev* 109: 545–572, 2002. doi:10.1037/0033-295X.109.3.545.

Gallego JA, Perich MG, Miller LE, Solla SA. Neural manifolds for the control of movement. *Neuron* 94: 978–984, 2017. doi:10.1016/j.neuron.2017.05.025.

Gao P, Ganguli S. On simplicity and complexity in the brave new world of large-scale neuroscience. *Curr Opin Neurobiol* 32: 148–155, 2015. doi:10.1016/j.conb.2015.04.003.

He K, Zhang X, Ren S, Sun J. Identity mappings in deep residual networks. In: *European Conference on Computer Vision: Lecture Notes in Computer Science*, edited by Leibe B, Matas J, Sebe N, Welling M. Cham, Switzerland: Springer, 2016, p. 630–645.

Hennequin G, Vogels TP, Gerstner W. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron* 82: 1394–1406, 2014. doi:10.1016/j.neuron.2014.04.045.

Kao JC, Nuyujukian P, Ryu SI, Churchland MM, Cunningham JP, Shenoy KV. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. *Nat Commun* 6: 7759, 2015. doi:10.1038/ncomms8759.

Kaufman MT, Churchland MM, Ryu SI, Shenoy KV. Cortical activity in the null space: permitting preparation without movement. *Nat Neurosci* 17: 440–448, 2014. doi:10.1038/nn.3643.

Kaufman MT, Churchland MM, Shenoy KV. The roles of monkey M1 neuron classes in movement preparation and execution. *J Neurophysiol* 110: 817–825, 2013. doi:10.1152/jn.00892.2011.

Kaufman MT, Seely JS, Sussillo D, Ryu SI, Shenoy KV, Churchland MM. The largest response component in motor cortex reflects movement timing but not movement type. *eNeuro* 3: ENEURO.0085-16, 2016. doi:10.1523/ENEURO.0085-16.2016.

Kingma DP, Ba J. Adam: a method for stochastic optimization. *International Conference on Learning Representations*. San Diego, CA, May 7–9, 2015.

Kleinman M, Chandramouli C, Kao J. A multi-stage recurrent neural network better describes decision-related activity in dorsal premotor cortex (Poster). *2019 Conference on Cognitive Computational Neuroscience*. Berlin, Germany, September 13–16, 2019.

Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*, edited by Pereira F, Burges CJC, Bottou L, Weinberger KQ. Lake Tahoe, NV: Curran Associates, p. 1097–1105, 2012.

Lukashin AV, Wilcox GL, Georgopoulos AP. Modeling of directional operations in the motor cortex: a noisy network of spiking neurons is trained to generate a neural-vector trajectory. *Neural Netw* 9: 397–410, 1996. doi:10.1016/0893-6080(95)00138-7.

Mante V, Sussillo D, Shenoy KV, Newsome WT. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* 503: 78–84, 2013. doi:10.1038/nature12742.

Michaels JA, Dann B, Scherberger H. Neural population dynamics during reaching are better explained by a dynamical system than representational tuning. *PLOS Comput Biol* 12: e1005175, 2016. doi:10.1371/journal.pcbi.1005175.

Miconi T. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife* 6: e20899, 2017. doi:10.7554/eLife.20899.

Nashed JY, Crevecoeur F, Scott SH. Influence of the behavioral goal and environmental obstacles on rapid feedback responses. *J Neurophysiol* 108: 999–1009, 2012. doi:10.1152/jn.01089.2011.

Nocedal J, Wright S. *Numerical Optimization*. New York: Springer, 2006.

Omrani M, Pruszynski JA, Murnaghan CD, Scott SH. Perturbation-evoked responses in primary motor cortex are modulated by behavioral context. *J Neurophysiol* 112: 2985–3000, 2014. doi:10.1152/jn.00270.2014.

Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. 30th International Conference on Machine Learning. Atlanta, GA, June 16–21, 2013.

Remington ED, Narain D, Hosseini EA, Jazayeri M. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron* 98: 1005–1019.e5, 2018. doi:10.1016/j.neuron.2018.05.020.

Sadtler PT, Quick KM, Golub MD, Chase SM, Ryu SI, Tyler-Kabara EC, Yu BM, Batista AP. Neural constraints on learning. *Nature* 512: 423–426, 2014. doi:10.1038/nature13665.

Santhanam G, Yu BM, Gilja V, Ryu SI, Afshar A, Sahani M, Shenoy KV. Factor-analysis methods for higher-performance neural prostheses. *J Neurophysiol* 102: 1315–1330, 2009. doi:10.1152/jn.00097.2009.

Sergio LE, Hamel-Pâquet C, Kalaska JF. Motor cortex neural correlates of output kinematics and kinetics during isometric-force and arm-reaching tasks. *J Neurophysiol* 94: 2353–2378, 2005. doi:10.1152/jn.00989.2004.

Song HF, Yang GR, Wang XJ. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLOS Comput Biol* 12: e1004792, 2016. doi:10.1371/journal.pcbi.1004792.

Song HF, Yang GR, Wang XJ. Reward-based training of recurrent neural networks for cognitive and value-based tasks. *eLife* 6: e21492, 2017. doi:10.7554/eLife.21492.

Stroud JP, Porter MA, Hennequin G, Vogels TP. Motor primitives in space and time via targeted gain modulation in cortical networks. *Nat Neurosci* 21: 1774–1783, 2018. [Erratum in *Nat Neurosci* 22: 504, 2019.] doi:10.1038/s41593-018-0276-0.

Sussillo D, Barak O. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput* 25: 626–649, 2013. doi:10.1162/NECO_a_00409.

Sussillo D, Churchland MM, Kaufman MT, Shenoy KV. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat Neurosci* 18: 1025–1033, 2015. doi:10.1038/nn.4042.

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015: 1–9, 2015. doi:10.1109/CVPR.2015.7298594.

Tanji J, Evarts EV. Anticipatory activity of motor cortex neurons in relation to direction of an intended movement. *J Neurophysiol* 39: 1062–1068, 1976. doi:10.1152/jn.1976.39.5.1062.

Townsend J, Koep N, Weichwald S. Pymanopt: a Python toolbox for optimization on manifolds using automatic differentiation. *J Mach Learn Res* 17: 1–5, 2016.

Weinrich M, Wise SP, Mauritz KH. A neurophysiological study of the premotor cortex in the rhesus monkey. *Brain* 107: 385–414, 1984. doi:10.1093/brain/107.2.385.

Yang GR, Joglekar MR, Francis Song H, Newsome WT, Wang XJ. Task representations in neural networks trained to perform many cognitive tasks. *Nat Neurosci* 22: 297–306, 2019. doi:10.1038/s41593-018-0310-2.

Yu BM, Cunningham JP, Santhanam G, Ryu SI, Shenoy KV. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *J Neurophysiol* 102: 614–635, 2009. doi:10.1152/jn.90941.2008.