nature methods

SUPPLEMENTARY INFORMATION

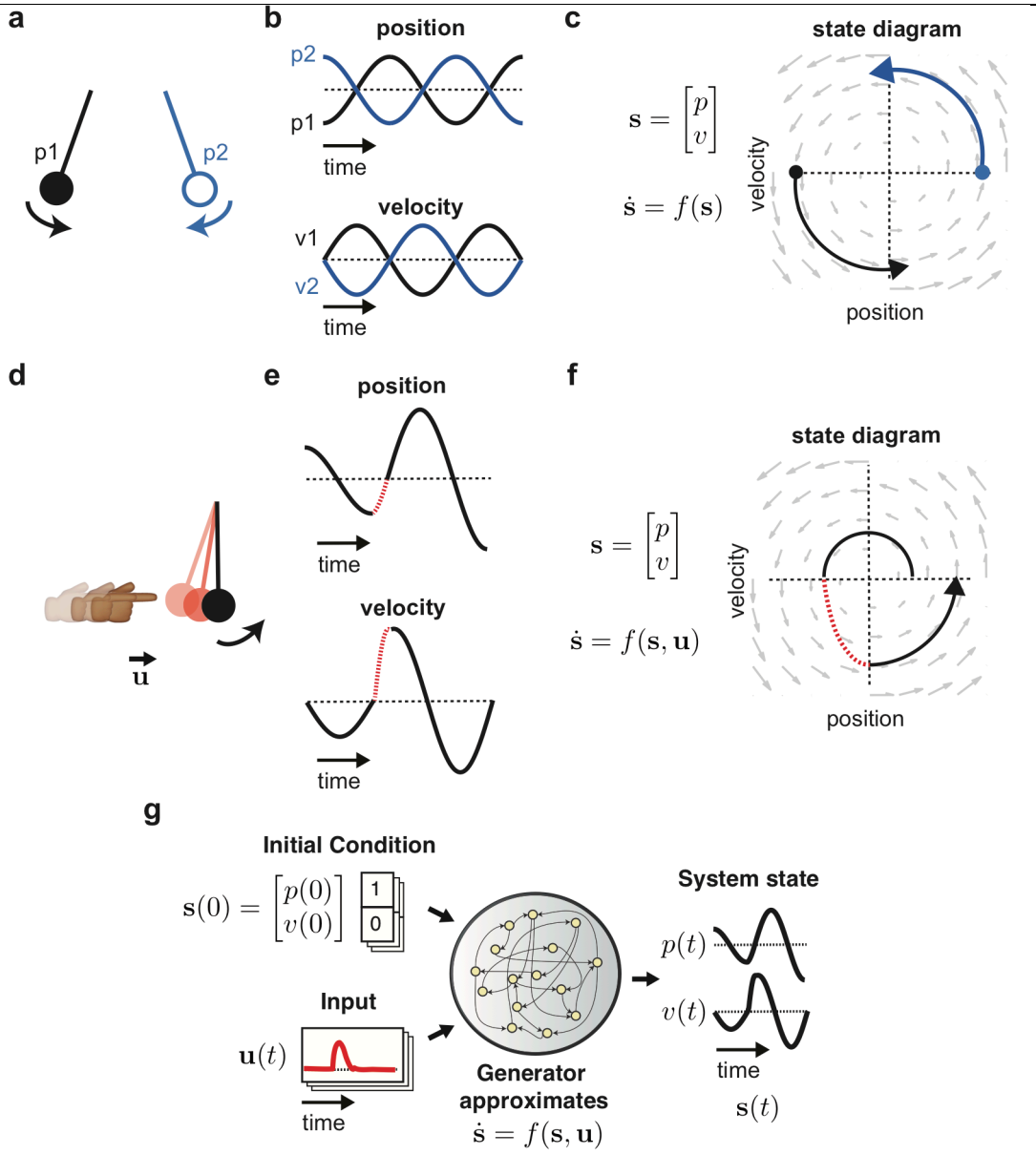
https://doi.org/10.1038/s41592-018-0109-9

In the format provided by the authors and unedited.

Inferring single-trial neural population dynamics using sequential auto-encoders

Chethan Pandarinath[1,2,3,4,5*], Daniel J. O'Shea[4,6], Jasmine Collins[7,20], Rafal Jozefowicz[7,21], Sergey D. Stavisky[3,4,5,6], Jonathan C. Kao[4,8], Eric M. Trautmann[6], Matthew T. Kaufman[6,22], Stephen I. Ryu[4,9], Leigh R. Hochberg[10,11,12], Jaimie M. Henderson[3,5], Krishna V. Shenoy[4,5,13,14,15,16], L. F. Abbott[17,18,19] and David Sussillo[4,5,7*]

[1]Wallace H. Coulter Department of Biomedical Engineering, Emory University and Georgia Institute of Technology, Atlanta, GA, USA. [2]Department of Neurosurgery, Emory University, Atlanta, GA, USA. [3]Department of Neurosurgery, Stanford University, Stanford, CA, USA. [4]Department of Electrical Engineering, Stanford University, Stanford, CA, USA. [5]Stanford Neurosciences Institute, Stanford University, Stanford, CA, USA. [6]Neurosciences Graduate Program, Stanford University, Stanford, CA, USA. [7]Google AI, Google Inc., Mountain View, CA, USA. [8]Department of Electrical Engineering, University of California, Los Angeles, Los Angeles, CA, USA. [9]Department of Neurosurgery, Palo Alto Medical Foundation, Palo Alto, CA, USA. [10]VA RR&D Center for Neurorestoration and Neurotechnology, Veterans Affairs Medical Center, Providence, RI, USA. [11]Center for Neurotechnology and Neurorecovery, Department of Neurology, Massachusetts General Hospital, Harvard Medical School, Boston, MA, USA. [12]School of Engineering and Carney Institute for Brain Science, Brown University, Providence, RI, USA. [13]Department of Neurobiology, Stanford University, Stanford, CA, USA. [14]Department of Bioengineering, Stanford University, Stanford, CA, USA. [15]Bio-X Program, Stanford University, Stanford, CA, USA. [16]Howard Hughes Medical Institute, Stanford University, Stanford, CA, USA. [17]Zuckerman Mind Brain Behavior Institute, Columbia University, New York, NY, USA. [18]Department of Neuroscience, Columbia University, New York, NY, USA. [19]Department of Physiology and Cellular Biophysics, Columbia University, New York, NY, USA. [20]Present address: University of California, Berkeley, Berkeley, CA, USA. [21]Present address: OpenAI, San Francisco, CA, USA. [22]Present address: Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, USA. *e-mail: chethan@gatech.edu; sussillo@google.com

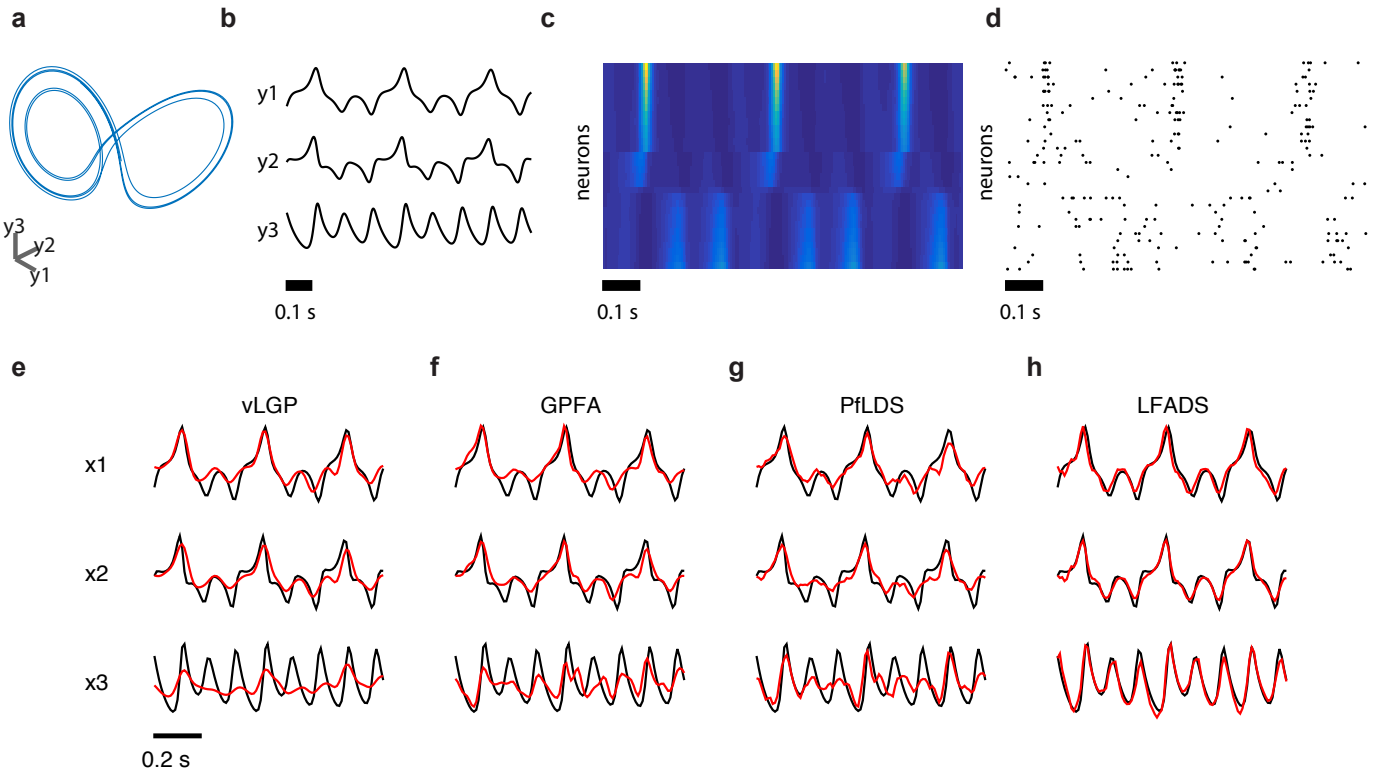


Supplementary Figure 1

Conceptual dynamical system: 1-D pendulum

a. A 1-dimensional pendulum released from position *p1* or *p2* has different positions and velocities over time. **b**. The state of the pendulum is captured by a 2-D vector that specifies its position and velocity, shown here evolving as a function of time (blue and black traces correspond to different initial conditions *p1* and *p2*). **c**. The evolution of the state follows dynamical rules, i.e. the pendulum's equations of motion. In this autonomous dynamical system (i.e., there are no perturbations), knowing the pendulum's initial state (filled circles) and dynamical rules that govern the state evolution (gray vector field) gives full knowledge of the pendulum's state as a function of time (black and blue traces). **d-f**. Perturbations to the pendulum, an example of input-driven dynamics. **d.** The pendulum's motion might be perturbed by an external force, e.g. it is bumped rightward. **e,f**. With a perturbation, the evolution of the system's state during the perturbation no longer follows its autonomous dynamical rules. This is shown as dashed red lines in the position vs. time and velocity vs. time plots, as well as the state-space diagram. A perturbation can be modeled by transforming the equations to allow an input term **u**(t) that models the perturbation. **g**. LFADS modeling of the perturbed pendulum. Traces of the pendulum motion are used to train LFADS. During training, the LFADS generator RNN learns to approximate the pendulum dynamics, $\dot{\mathbf{s}} = f(\mathbf{s}, \mathbf{u})$, using its own internal state and dynamics. LFADS also learns a per-trial initial generator state s0, which allows it to model trials that start with different initial pendulum states, such as *p1* or *p2*. Further, LFADS learns a set of time-varying inputs per trial, which allows it to model

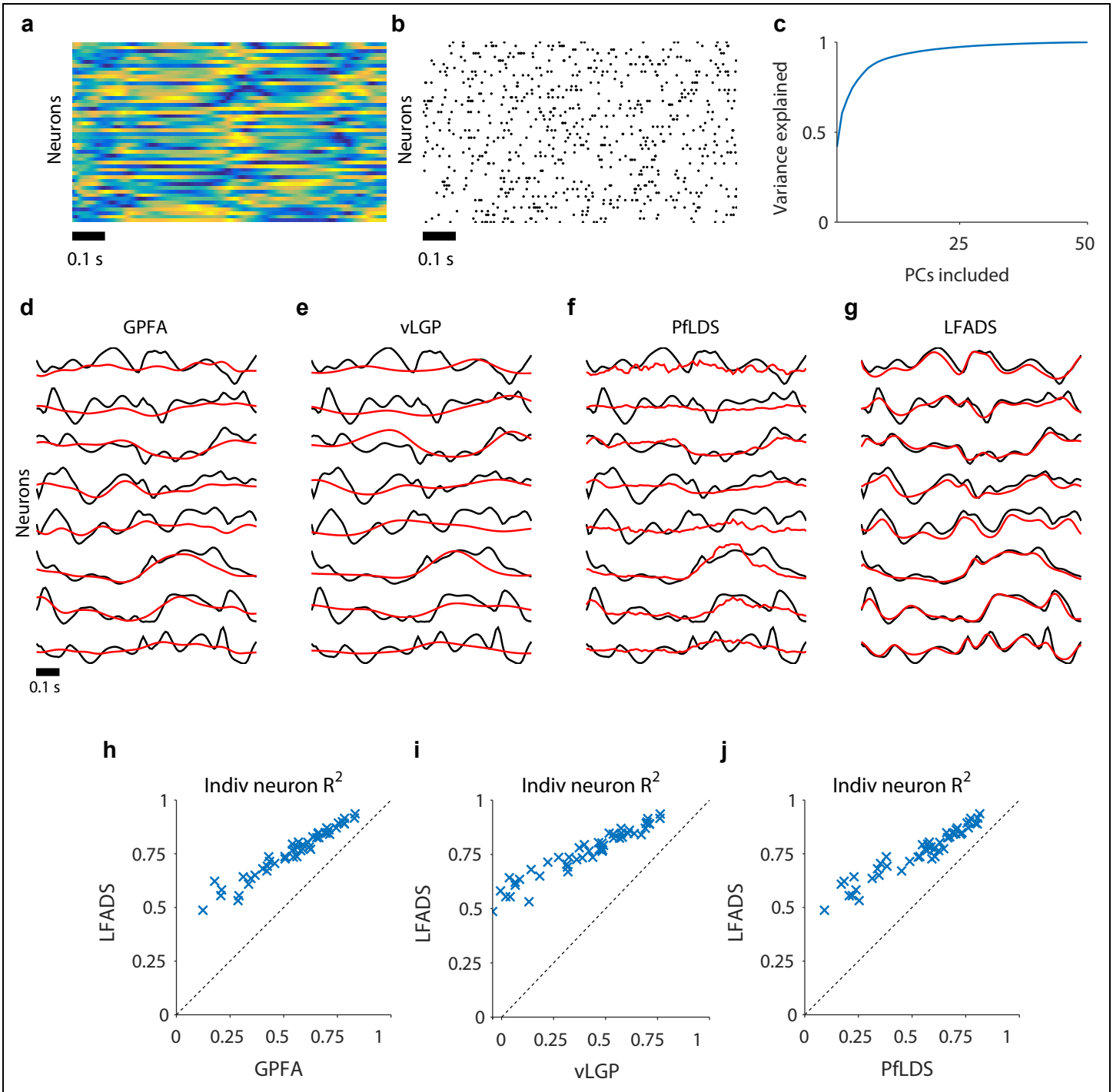
perturbations to the pendulum system, **u**(t). These three pieces of information are enough to reconstruct each trial.



Supplementary Figure 2

LFADS applied to Lorenz attractor

We compared the performance of LFADS to three existing methods that estimate latent state from neural data: Variational Latent Gaussian Process (vLGP), Gaussian Process Factor Analysis (GPFA), and Poisson Feed-forward neural network Linear Dynamical System (PfLDS). To test LFADS and to compare its performance with other approaches, we generated synthetic stochastic spike trains from deterministic nonlinear systems. The first is the standard Lorenz system (see Online Methods for equations and details). **a**. An example trial illustrating the evolution of the Lorenz system in its 3-dimensional state space, and **b.** its dynamic variables as a function of time. **c**. Firing rates for the 30 simulated neurons are generated by a linear readout of the latent variables followed by an exponential nonlinearity, with neurons sorted according to their weighting for the first Lorenz dimension. **d**. Spike times for the neurons are generated from the rates of the simulated neurons. **e-h** Sample performance for each method applied to spike trains based on Lorenz attractor. Each panel shows actual (black) and inferred (red) values of the three latent variables for a single example trial for the 4 methods: **e**. vLGP, **f**. GPFA, **g**. PfLDS, **h.** LFADS. For LFADS, posterior means were averaged over 128 samples of **g0** conditioned on the particular data sequence being decoded. We quantified performance using R^2, i.e., the fraction of the variance of the actual latent variables captured by the estimated latent values. For the latent dimensions {y1,y2,y3}, the resulting R^2 values were {0.761,0.688,0.218}, {0.713,0.725,0.325}, {0.784,0.732,0.368}, and {0.850,0.921,0.872} for vLGP, GPFA, PfLDS, and LFADS, respectively.

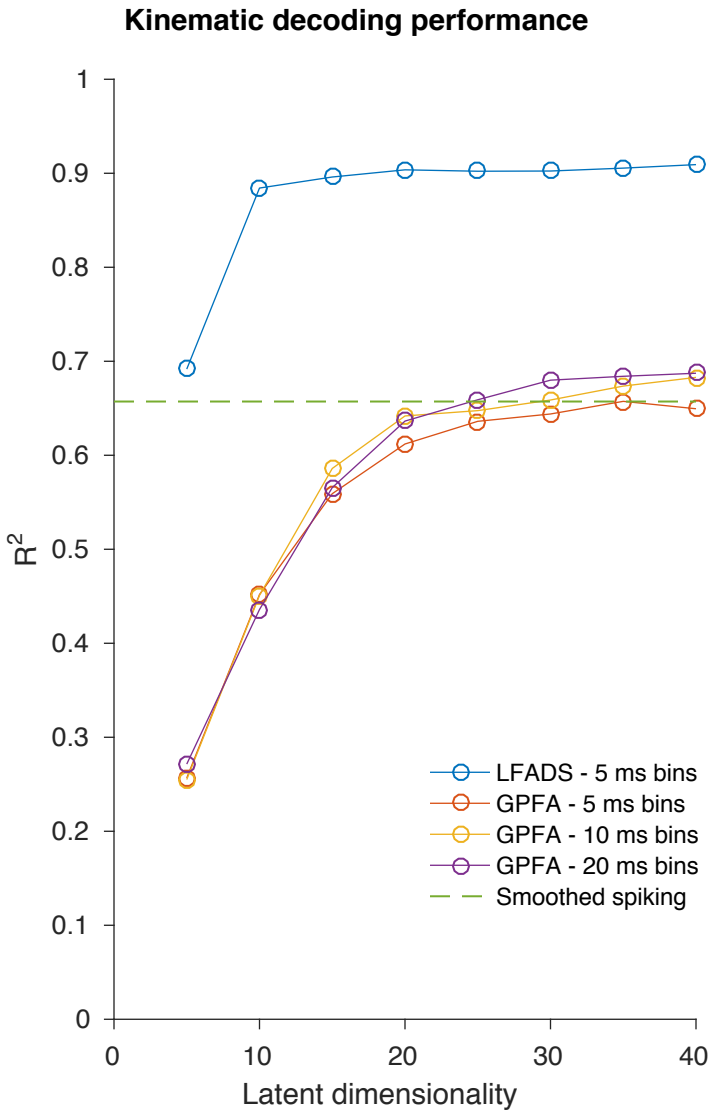


Supplementary Figure 3

LFADS applied to autonomous chaotic data RNN

a-c. We generated synthetic data with high-dimensional, chaotic dynamics using an RNN. **a**. Firing rates for one example trial, simulated by the chaotic "data RNN" (colors show rates fluctuating between -1 and 1). **b**. We then created synthetic spike trains, emitted from a Poisson process whose underlying rates were the normalized continuous rates of the data RNN. **c.** We used principal components analysis to assess the dimensionality of the data. As expected, the state of the data RNN had lower dimension than its

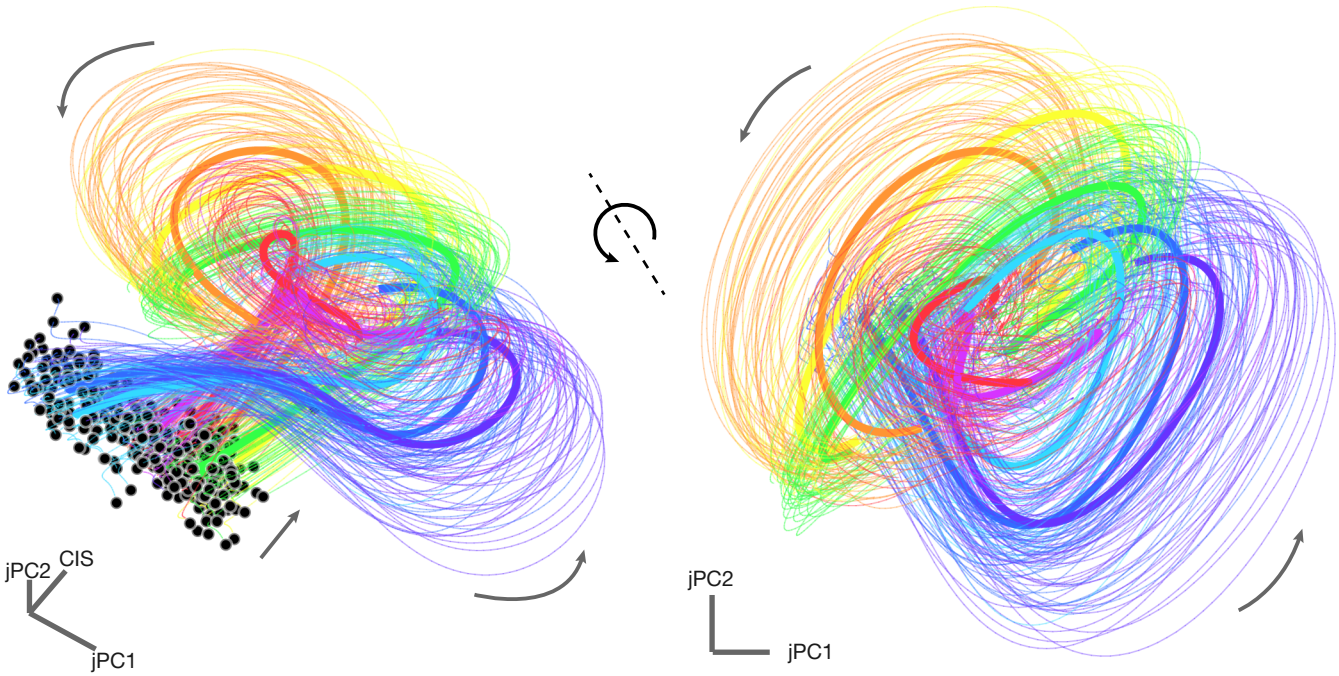
number of neurons, and 20 principal components were sufficient to capture > 95% of the variance of the system. So we restricted the latent space to 20 dimensions for each of the models tested and, in the case of LFADS, set the factors dimensionality to 20 as well (F = 20). **d-g** Sample performance for each method on the RNN task. We tested the performance of the methods at extracting the underlying firing rates from the spike trains of the RNN dataset. Shown are single trial examples for **d** GPFA, **e** vLGP, **f** PfLDS, and **g** LFADS. As can be seen by eye, the LFADS results are closer to the actual underlying rates than for the other models (black, firing rates of chaotic data RNN, red, inferred rates). **h-j**. Summary R^2 values between actual and inferred rates. Comparison using held-out data of the R2 values for **h** GPFA vs. LFADS, **i** vLGP vs. LFADS, and **j** PfLDS vs. LFADS for all units (blue 'x'). In all comparisons, LFADS yields a better fit to the data, for every single unit.



Supplementary Figure 4

Effect of varying hyperparameters on kinematic decoding performance in the maze task

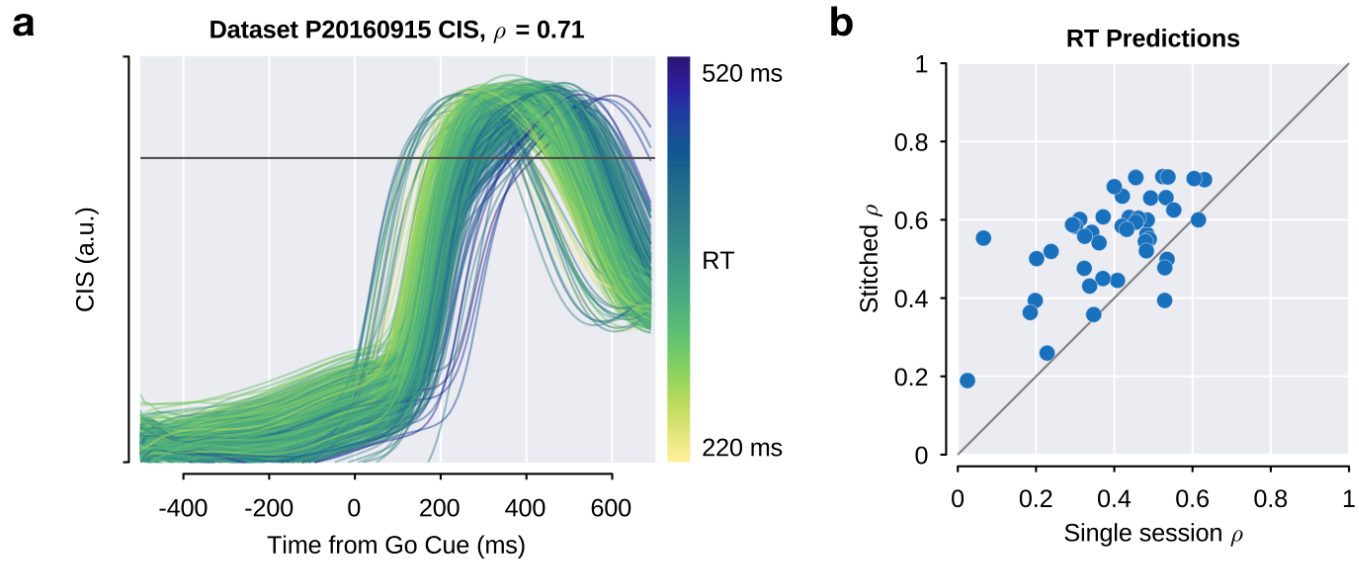
For GPFA, we tested the effect of varying binsize and latent dimensionality on the ability to decode arm velocities. For LFADS, we fixed the binsize at 5 ms, and tested the effect of changing the latent dimensionality (number of factors).



Supplementary Figure 5

Single-trial factor trajectories from the multi-session stitched LFADS model for a representative session

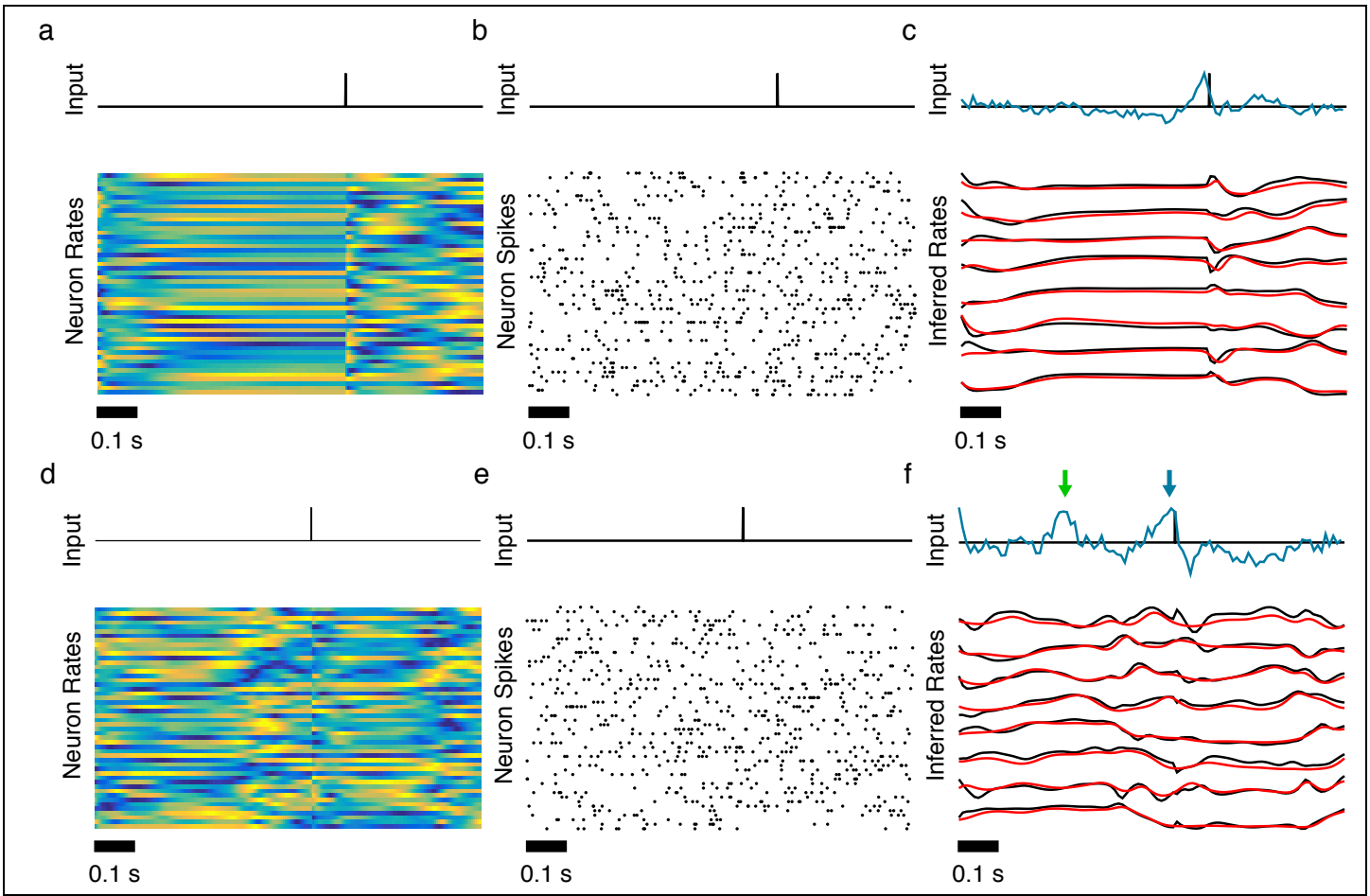
LFADS factors are projected into a space consisting of the condition-independent signal (CIS) and the first rotational plane identified using jPCA; this is the same projection used in **Fig. 4d** in the main text. Black dots depict the time of go cue presentation, and each trajectory proceeds for 510 ms. Colors indicate reach direction as in **Fig. 4**. Thin traces are trajectories from individual trials; thick traces of the same color are the condition-averaged trajectory for each reach direction.



Supplementary Figure 6

Dynamically stitched multi-session LFADS model outperforms single-session LFADS models in predicting reaction times

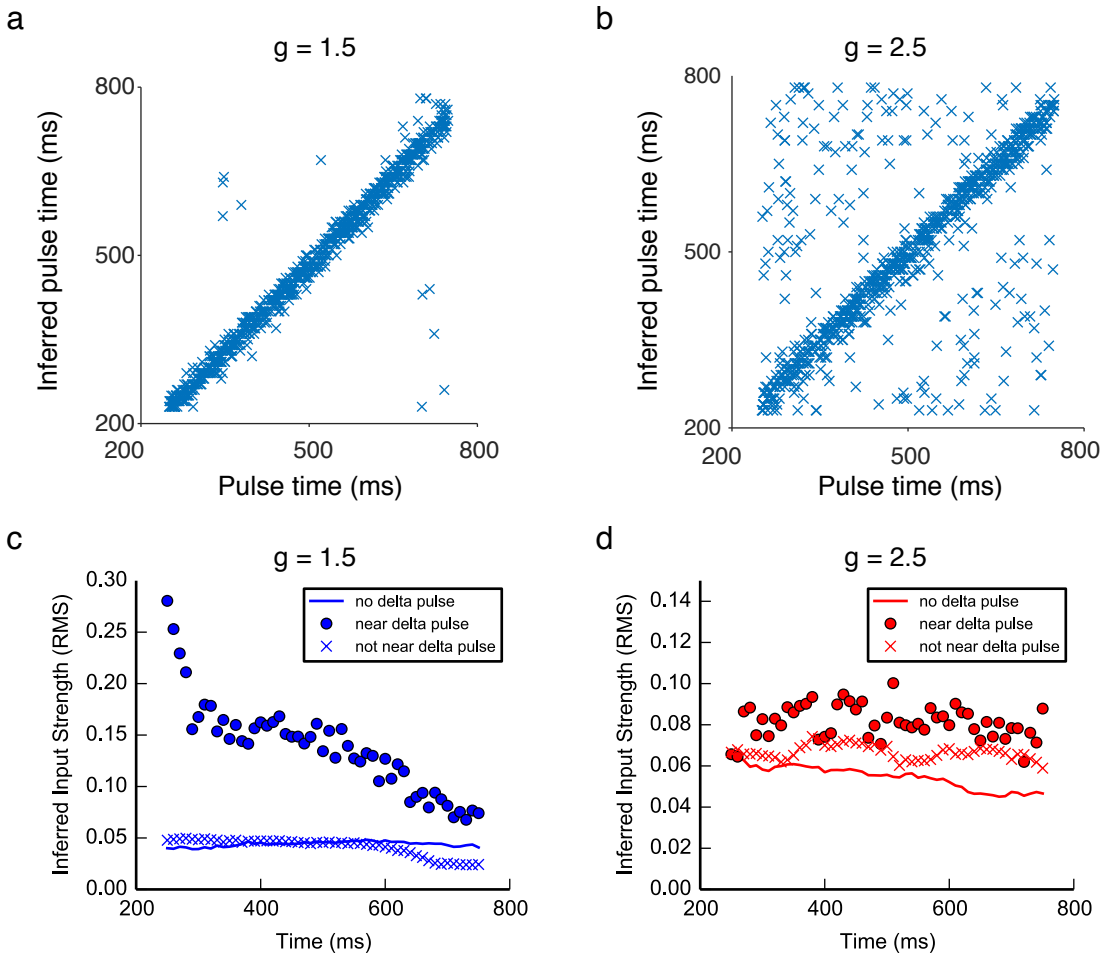
As defined in (Kaufman et al., *eNeuro* 2016), the condition-independent signal (CIS) is a high variance component of motor cortical population activity obtained via demixed principal components analysis (dPCA). Kaufman et al. 2016 also demonstrated that threshold crossing time of the CIS on single trials is an effective predictor of reach reaction time (RT). Here we identify the CIS as a linear projection of LFADS factor trajectories. We apply dPCA to the factor outputs of each single-session and the multi-session LFADS models to identify the largest condition-independent component, and then threshold the CIS to predict RT on single trials. **a**. Plot of condition-independent signals (CIS) for an example dataset. Each trace represents the CIS timecourse on a single trial, and is colored by that trial's actual RT. **b**. Plot of correlations between CIS-predicted RT and actual RT on trials from each dataset for stitched multi-session LFADS vs single-session LFADS. Each point represents an individual recording session. For the stitched model, a single CIS projection was computed and applied for all sessions, whereas individual CIS projections were obtained for each single-session model.



Supplementary Figure 7

Inferring inputs from a chaotic data RNN with delta pulse inputs

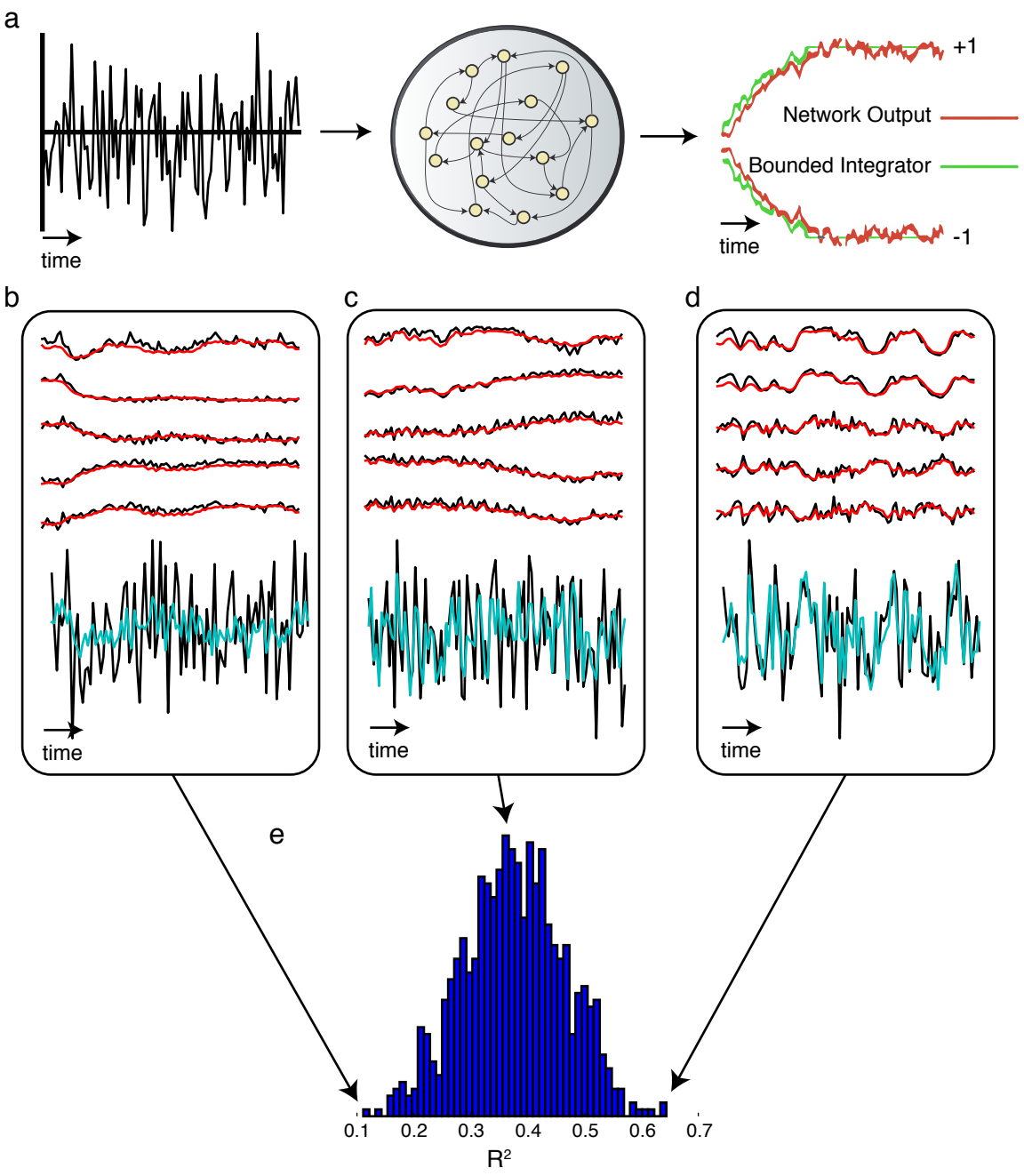
We tested the ability of LFADS to infer the input to a dynamical system, specifically chaotic data RNNs, as used in **Supp. Fig. 3**. During each trial, we perturbed the network by delivering a delta pulse at a random time t_{pulse} between 0.25s and 0.75s. The full trial length was 1s. This pulse affected the underlying rates produced by the data RNN, which subsequently affected the generated spike trains that were used as input to LFADS. To test the ability of LFADS to infer the timing of the input pulses, we allowed the LFADS model to infer a time-varying input (1-dimensional). We explored two levels of dynamical complexity in the data RNNs (see Online Methods), defined by two values, 1.5 and 2.5, of a hyper-parameter to the data RNN, γ. **a-c** γ = 1.5. This value of γ value produces “gentler" chaotic activity in the data RNN than the higher value. **a.** Example trial illustrating results from the γ = 1.5 chaotic data RNN with an external input (shown in black at the top of each column). Firing rates for the 50 simulated neurons. **b**. Poisson-generated spike times for the simulated neurons. **c**. Example trial showing (top) the actual (black) and inferred (cyan) input, and (bottom) actual firing rates of a subset of neurons in black and the corresponding inferred firing rates in red (bottom). **d-f**. Same as a-c, but for γ = 2.5, which produces significantly more chaotic dynamics than γ = 1.5. **f.** For this more difficult case, LFADS inferred input at the correct time (blue arrow), but also used its 1-D inferred input to shape the dynamics at times there was no actual input (green arrow).



Supplementary Figure 8

Summary of results of chaotic data RNNs receiving input pulses

We extracted averaged inferred inputs, u_t, from the LFADS models used in **Supp. Fig. 7**. **a,b.** To see how timing of the inferred input was related to the timing of the actual input pulse, we determined the time at which u_t reached its maximum value (vertical axis), and plotted this against the actual time of the delta pulse (horizontal axis) for all trials. **a**. Results using γ = 1.5, **b**. and γ = 2.5. As shown, for the majority of trials, despite the complex internal dynamics, LFADS was able to infer the correct timing of a strong input. However, LFADS did a better job of inferring the inputs in the case of simpler dynamics for two reasons: In the case of γ = 2.5, 1) the complexity of the dynamics reduces the effective magnitude of the perturbation caused by the input and 2) LFADS used the inferred input more actively to account for non-input-driven dynamics. We include this example of a highly chaotic data RNN to highlight the subtlety of interpreting an inferred input. **c-d** One possibility in using LFADS with inferred inputs (i.e. dimensionality of $u_t \geq 1$) is that the data to be modeled is actually generated by an autonomous system, yet one, not knowing this fact, allows for an inferred input in LFADS. To study this case we utilized the four chaotic data RNNs described above, i.e. γ = 1.5, and γ = 2.5, with and without delta pulse inputs. We trained an LFADS model for each of the four cases, with an inferred input of dimensionality 1, despite the fact that two of the four data RNNs generated their data autonomously. After training we examined the strength of the average inferred input, ut, for each LFADS model (where strength is taken as the root-mean-square of the inferred input, averaged over an appropriate time window, $\mathrm{sqrt}(\langle u^2_t \rangle_{t_1:t_2})$). The results are show in panel **c** for γ = 1.5 and panel **d** for γ = 2.5. The solid lines show the strength u_t at each time point when the data RNN received no input pulses, averaged across all examples. The '◦' and 'x' show the strength of u_t at time points when the data RNN received delta pulses, averaged in a time window around t, and averaged over all examples. Intuitively, a '◦' is the strength of u_t around a delta pulse at time t, and an 'x' is the strength of u_t if there was no delta pulse around time t.

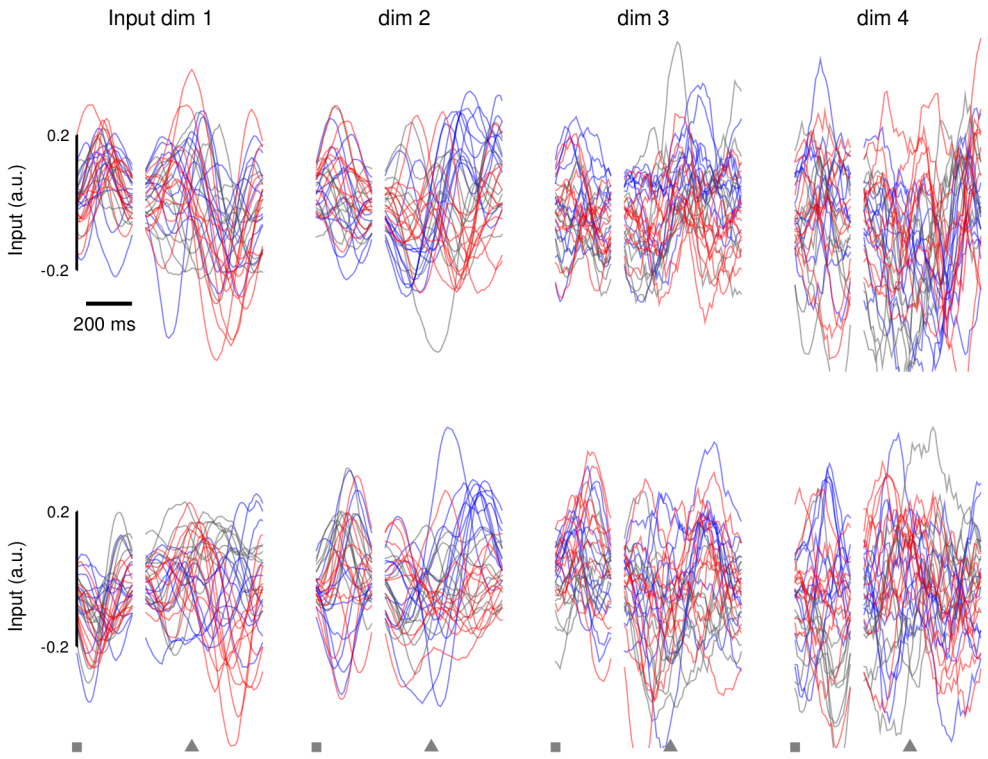


Supplementary Figure 9

Inferring input from a data RNN trained to integrate white noise to a bound

LFADS is able to model simulated neurons that are integrating a noisy input, and also infer the input itself (the noise signal). **a**. Overview of the integration-to-bound task. On each trial, the data RNN receives noise drawn from a Gaussian distribution with mean 0, variance 0.0625. We trained an RNN to integrate this stochastic, 1-dimensional input to either a high (+1) or low (−1) bound. After the data RNN learned the task, we generated spiking data from 50 neurons using similar methodology as **Supp. Fig. 3** and fit an LFADS model to this data. **b-d** We fit an LFADS model to the data using 3200 1-second training examples, and evaluated its performance on 800 held-out trials. LFADS was able to accurately infer the ground truth firing rates (LFADS in red, ground truth in black). LFADS also inferred the associated white-noise input to the data RNN (LFADS cyan, ground truth in black, posterior means averaged over 1024 samples). These panels show the trials with the worst, median, and best measured R^2 values between true and inferred inputs. **b.** Trial

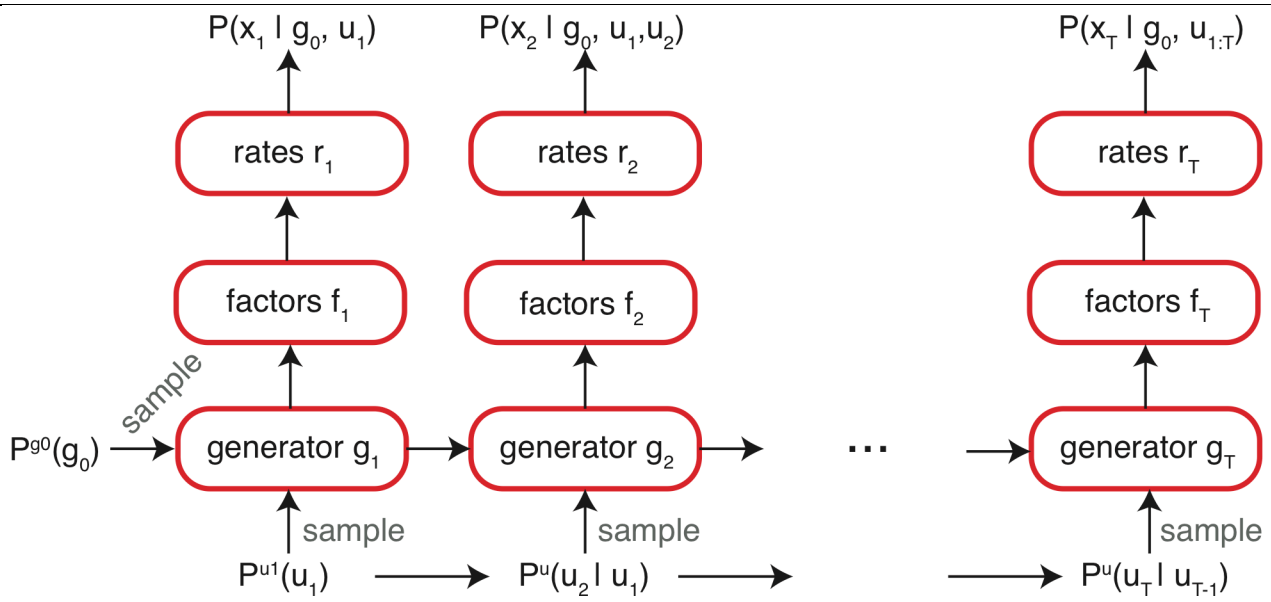
with worst R^2 = 0.11, **c.** median R^2 = 0.38, **d.** and the best R^2 = 0.64. **e.** Histogram showing distribution of R^2 values between true and inferred inputs for the 800 held-out trials.



Supplementary Figure 10

Inferred inputs from individual trials in the Cursor Jump task

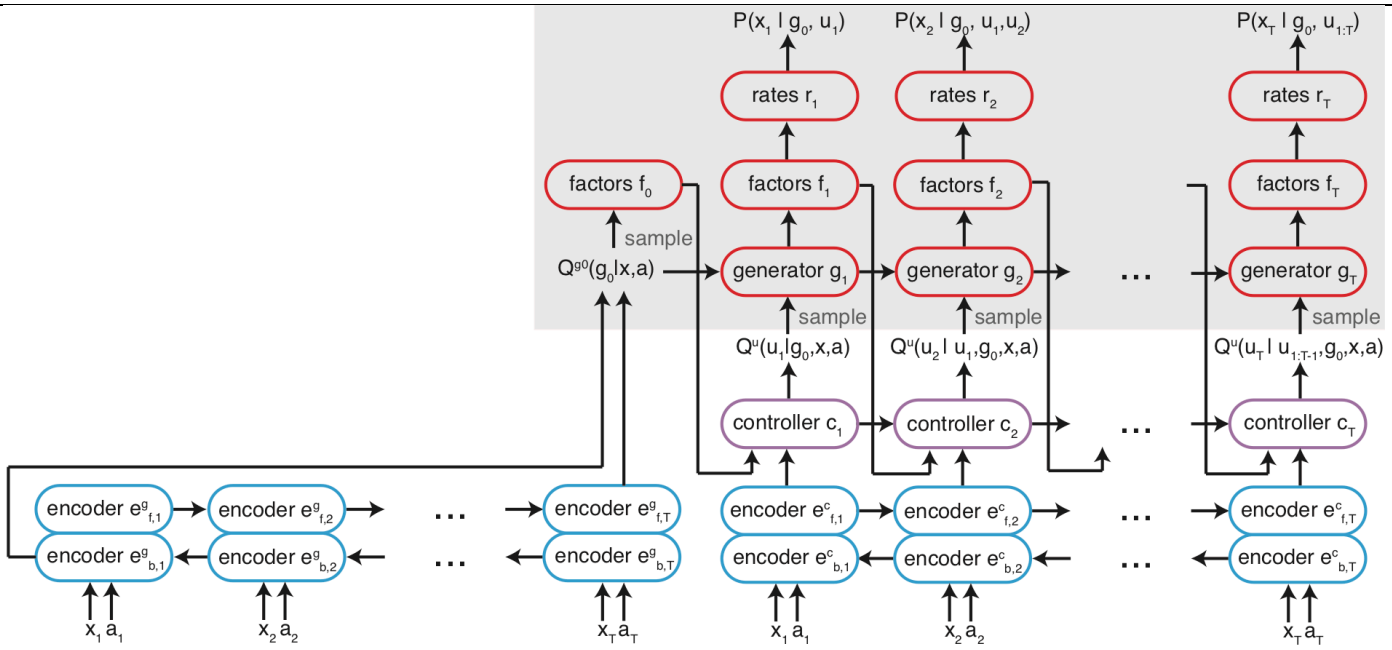
This figure parallels **Fig. 5d** from the main text, but in this case, we plot the inferred inputs for individual trials. To increase visibility, only 10 trials for each condition (reach direction and perturbation type) are shown. Individual traces were smoothed with an acausal Gaussian filter (60 ms s.d.). Despite the high variance across individual trials, several of the trends in the inferred inputs described in **Fig. 5** in the main text are visible at the single-trial level. The inputs show information about the target of the upcoming reach on a single-trial level, though individual traces are noisy. Specifically, for example, at the time of target onset (squares), the inferred input dimension 1 diverges for up vs. down reaches, but not for different perturbation types (as the information about perturbation type is not yet known at that phase of the task). Further, the inputs show information about the perturbation timing and identity on a single-trial level, though again, individual traces are noisy. Specifically, around the time of the perturbation (arrow), the traces diverge for left-perturbed vs. right-perturbed vs. unperturbed trials (e.g., seen in dimension 2). Though these individual traces are noisy, **Fig. 5e** in the main text shows that these inputs can largely be separated on a single-trial basis using a nonlinear dimensionality reduction algorithm, t-SNE.



Supplementary Figure 11

The LFADS generator

The generative LFADS model is a recurrent network with a feed-forward read-out. The generator takes a sampled initial condition, $\hat{\mathbf{g}}_0$ and a sampled inferred input, $\hat{\mathbf{u}}_t$, at each time step, and iterates forward. At each time step the temporal factors, $\mathbf{f}_t$, and the rates, $\mathbf{r}_t$ are generated in a feed-forward manner from $\mathbf{g}_t$. Spikes are generated from a Poisson process, $\hat{\mathbf{x}}_t \sim \text{Poisson}(\mathbf{x}_t|\mathbf{r}_t)$. The initial condition and input at time step 1 are sampled from diagonal Gaussian distributions with zero mean and fixed chosen variance. Otherwise, the inputs are sampled from a Gaussian auto-regressive prior.



Supplementary Figure 12

The full LFADS model for inference

The generator / decoder portion highlighted with a gray background and is colored red, the encoder portion is colored blue and the controller, purple. To infer the latent dynamics from the recorded neural spike trains $\mathbf{x}_{1:T}$ and conditioning data $\mathbf{a}_{1:T}$, initial conditions for the controller and generator networks are encoded from inputs. In the case of the generator, the initial condition $\hat{\mathbf{g}}_0$ is drawn from an approximate posterior $Q^{g_0}(\mathbf{g}_0|\mathbf{x}_{1:T}, \mathbf{a}_{1:T})$ that receives an encoding of the input, $\mathbf{E}^{gen}$ (in this figure, for compactness, we use $\mathbf{x}$ and $\mathbf{a}$ to denote $\mathbf{x}_{1:T}$ and $\mathbf{a}_{1:T}$). The low-dimensional factors at $t = 0$, $\mathbf{f}_0$, are computed from $\hat{\mathbf{g}}_0$. The controller then propagates one step forward in time, receiving the sample factors $\mathbf{f}_0$ as well as bidirectionally encoded inputs $\mathbf{E}_1^{con}$ computed from $\mathbf{x}_{1:T}, \mathbf{a}_{1:T}$. The controller produces, through an approximate posterior $Q^u(\mathbf{u}_1|\mathbf{g}_0, \mathbf{x}_{1:T}, \mathbf{a}_{1:T})$, a sampled inferred input $\hat{\mathbf{u}}_1$ that is fed into the generator network. The generator network then produces $\{\mathbf{g}_1, \mathbf{f}_1, \mathbf{r}_1\}$, with $\mathbf{f}_1$ the factors, and $\mathbf{r}_1$ the Poisson rates at $t = 1$. The process continues iteratively so, at time step t, the generator network receives $\mathbf{g}_{t-1}$ and $\hat{\mathbf{u}}_t$ sampled from $Q^u(\mathbf{u}_t|\mathbf{u}_{1:t-1}, \mathbf{g}_0, \mathbf{x}_{1:T}, \mathbf{a}_{1:T})$. The job of the controller is to produce a nonzero inferred input only when the generator network is incapable of accounting for the data autonomously. Although the controller is technically part of the encoder, it is run in a forward manner along with the decoder.

Model	Figure	N	F	$\|\mathbf{u}_t\|$	g0 E dim	$\mathbf{u}_t$ E dim	C dim	G L_2	C L_2	KP	BS
Monkey J Maze	Main 2,3	100	40	0	100	-	-	10	-	0.98	5
Participant T5 Center-out	Main 3	64	20	3	64	-	-	250	-	0.95	5
Monkey P Multi-session	Main 4	100	16	0	100	-	-	500	-	0.98	10
Monkey P Single-session	Main 4	100	16	0	100	-	-	500	-	0.98	10
Monkey J CursorJump	Main 5	128	50	4	150	100	128	25	25	0.98	10
Monkey J Center-out	Main 6	128	50	4	150	100	128	25	25	0.98	2
Participant T7 Center-out	Main 6	64	20	3	64	64	128	250	250	0.95	5
Lorenz attractor	Supp. 2	64	3	0	64	-	-	250	-	0.95	a.u.
Chaotic RNN	Supp. 3	200	20	0	200	-	-	2000	-	0.95	a.u.
Input pulses	Supp. 6,7	200	20	1	200	128	128	2000	0	0.95	a.u.
RNN Integrator	Supp. 8	200	20	1	128	128	128	2000	0	0.95	a.u.

Supplementary Table 1. Important hyper-parameters of LFADS models. Listed here are the most important LFADS parameters, relating primarily to model capacity. ’N’ - number of units in the generator, ’F’ - number of factors, $|\mathbf{u}_t|$ - number of inferred inputs, ’E’ - encoder, ’C’ - controller, ’G’ - generator, ’KP’ - keep probability in dropout layers, ’BS’ - bin size (ms).

Model	Figure	Electrode type	Signal post-processing
Monkey J Maze	Main 1, 2, 3	Utah array	threshold crossings, spike sorted
Participant T5 Center-out	Main 3	Utah array	threshold crossing
Monkey P Single-session	Main 4	v-probe	threshold crossing
Monkey P Multi-session	Main 4	v-probe	threshold crossing
Monkey J CursorJump	Main 5	Utah array	threshold crossing
Monkey J Center-out	Main 6	Utah array	threshold crossing
Participant T7 Center-out	Main 6	Utah array	threshold crossing

Supplementary Table 2. Signal collection technology and spike detection methods.

Supplementary Note 1: Synthetic datasets

Summary of synthetic datasets

We chose a variety of synthetic examples in an effort to show LFADS's ability to infer informative representations for dynamical systems of varying complexity. We ordered the synthetic examples roughly by complexity to build intuition. The examples are, in order,

1. The pendulum example (**Supp. Fig. 1**) - a cartoon (no actual data), simply intended to impart intuition using a well-known and tangible physical system.

2. The Lorenz model (**Supp. Fig. 2, Supp. Table 1**) - this simple model is now becoming standard in the field (e.g., [1,2]), as it is a simple and well-known example of a nonlinear, chaotic dynamical system, and easy to understand and visualize due to its 3D state space.

3. A synthetic RNN example with random connections and without input (**Supp. Fig. 3**) - this creates a much more complex high-dimensional dynamical system, intended to differentiate our method from common methods in the field that have difficulty modeling high-dimensional, highly nonlinear dynamics. This RNN does not have the same architecture as that used in LFADS.

4. A synthetic RNN example with simple pulse inputs (**Supp. Figs. 7,8**) - this provides a clear demonstration of the ability of LFADS to decompose an observed time series into both dynamics and inputs. This RNN does not have the same architecture as that used in LFADS.

5. A synthetic RNN trained to perform an integration-to-bound task, given a noisy 1-D input (**Supp. Fig. 9**). Integration-to-bound is a common model of decision-making in systems neuroscience. This example shows the utility of LFADS not only in modeling a network that is trained to perform a task, but also shows that LFADS can infer inputs in networks that are performing meaningful computations. This RNN does not have the same architecture as that used in LFADS.

Lorenz system

The Lorenz system is a set of nonlinear equations for three dynamic variables. Its limited dimensionality allows its entire state space to be visualized. The evolution of the system's state is governed as follows

$$\dot{y}_1 = \sigma(y_2 - y_1) \quad (1)$$
$$\dot{y}_2 = y_1(\rho - y_3) - y_2 \quad (2)$$
$$\dot{y}_3 = y_1 y_2 - \beta y_3. \quad (3)$$

We used the standard parameter values known for inducing chaos, $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$, and used Euler integration with $\Delta t = 0.006$. As in [1], we simulated a population of neurons with firing rates given by linear read-outs of the Lorenz variables using random weights, followed by an exponential nonlinearity. Spikes from these firing rates were then generated by a Poisson process.

Our synthetic dataset consisted of 65 conditions, with 20 trials per condition. Each condition was obtained by starting the Lorenz system with a random initial state vector and running it for 1s.

Twenty different spike trains were then generated from the firing rates for each condition. Models were trained using 80% of the data (16 trials/condition) and evaluated using 20% of the data (4 trials/condition). While this simulation is structurally quite similar to the Lorenz system used in [1], we purposefully chose parameters that made the dataset more challenging. Specifically, relative to [1], we limited the number of observations to 30 simulated neurons instead of 50, decreased the baseline firing rate from 15 spikes/sec to 5 spikes/sec, and sped up the dynamics by a factor of 4.

Chaotic RNNs as data generators

We tested the performance of each method at inferring the dynamics of a more complex nonlinear dynamical system, a fully recurrent nonlinear neural network with strong coupling between the units. We generated a synthetic dataset from an N-dimensional continuous time nonlinear, so-called, "vanilla" RNN,

$$\tau\, \dot{\mathbf{y}}(t) \quad = -\mathbf{y}(t) + \gamma\, \mathbf{W}^{y} \tanh(\mathbf{y}(t)) + \mathbf{B}\, \mathbf{q}(t). \quad (40)$$

This makes a compelling synthetic case study for our method because many recent studies of neuroscientific data have used vanilla RNNs as their modeling tool (e.g. [3–7]). It should be stressed that the vanilla RNN used as the data RNN here does not have the same functional form as the network generator used in the LFADS framework, which is a GRU (see section *1.7*), although both have continuous variables and are not spiking models. For experiments in **Supp. Fig. 3**, we set $\mathbf{B} = \mathbf{q} = 0$, but we included an input for experiments in **Supp. Fig. 6**.

The elements of the matrix $\mathbf{W}^{y}$ were drawn independently from a normal distribution with zero mean and variance $1/N$. We set γ to either 1.5 or 2.5, both of which produce chaotic dynamics at a relatively slow timescale compared to τ (see [3] for more details). The smaller γ value produces "gentler" chaotic activity in the data RNN than the larger value. Specifically, we set $N = 50$, $\tau = 0.025$ s and used Euler integration with $\Delta t = 0.01$ s. Spikes were generated by a Poisson process with firing rates obtained by scaling each element of $\tanh(\mathbf{y}(t))$ to take values in $[0,1]$, and then used as the rate in a Poisson process to give rates lying between 0 and 30 spikes/s.

Our dataset consisted of 400 conditions obtained by starting the data RNN at different initial states with elements drawn from a normal distribution with zero mean and unit variance. Firing rates were then generated by running the data RNN for 1 s, and 10 spiking trials were produced for each condition, yielding a total of 4,000 spiking trials. Models were trained using 80% of the data (8 trials/condition) and evaluated using 20% of the data (2 trials/condition).

Inferring pulse inputs to a chaotic RNN

We tested the ability of LFADS to infer the input to a chaotic RNN (**Supp. Figs. 6,7**). In general, the problem of disambiguating dynamics from inputs is ill-posed, so we encouraged the dynamics to be as simple as possible by including an L_2 regularizer in the LFADS network generator (see **Supplementary Table 1**). We note that weight regularization is a standard technique that is nearly universally applied to neural network architectures.

Focusing on **Supp. Fig 6**, we studied the synthetic example of inferring the timing of a delta pulse input to a randomly initialized RNN. To introduce an input into the data RNN, the elements of **B** were drawn independently from a normal distribution with zero mean and unit

variance. During each trial, we perturbed the network by delivering a delta pulse of magnitude 50, $q(t) = 50\delta(t - t_{pulse})$, at a random time t_{pulse} between 0.25s and 0.75s (the full trial length was 1s). This pulse affects the underlying rates produced by the data RNN, which modulates the spike generation process. To test the ability of the LFADS model to infer the timing of these input pulses, we included in the LFADS model an inferred input with dimensionality of 1. We explored the same two values of γ as in the synthetic example to model chaotic RNN dynamics, 1.5 and 2.5. Other than adding the input pulses, the data for input-pulse perturbations were generated as in the first data RNN example described above.

After training, which successfully inferred the firing rates, we extracted inferred inputs from the LFADS model (eqn. 15) by running the system 512 times for each trial, and averaging, defining $\overline{\mathbf{u}}_t = \langle \mathbf{u}_t \rangle_{\mathbf{g}_0, \mathbf{u}_{1:T}}$. To see how the timing of the inferred input was related to the timing of the actual input pulse, we determined the time at which $\overline{\mathbf{u}}_t$ reached its maximum value.

Inferring white noise input in an RNN trained to integrate to bound

We tested the ability of LFADS to infer the input to a vanilla RNN trained to integrate a noisy signal to a $+1$ or -1 bound. Weight matrices for this "data simulation RNN" were drawn independently from a Gaussian distribution with zero mean and variance $0.64/N$, and L_2 regularization was used during training. The noisy input signal was drawn from a Gaussian distribution with zero mean and variance 0.0625. 800 conditions were generated with white noise inputs, and 5 spiking trials were generated per condition. This resulted in 4,000 1s spiking trials. 3,200 trials were used for training and 800 trials were used for validation.

After training LFADS on the integrate-to-bound data (simulated as above), inferred inputs ($\overline{\mathbf{u}}_t$) for a given trial were extracted by taking 1024 samples from the ($\overline{\mathbf{u}}_t$) posterior distribution produced by LFADS, and then averaging. These inferred inputs were then compared (using R^2) with the real inputs to the integrate-to-bound model, which were saved down previously during training.

Supplementary Note 2: LFADS-related work in machine learning literature

Recurrent neural networks have been used extensively to model neuroscientific data (e.g. [3–7]), but the networks in these studies were all trained in a deterministic setting. An important recent development in deep learning has been the advent of the variational auto-encoder [8,9], which combines a probabilistic framework with the power and ease of optimization of deep learning methods. VAEs have since been generalized to the recurrent setting, for example with variational recurrent networks[10], deep Kalman filters[11], and the RNN DRAW network[12].

There is also a line of research applying probabilistic sequential graphical models to neural data. Recent examples include PLDS[13], switching LDS[14], GCLDS[15], and PfLDS[16]. These models employ a linear Gaussian dynamical system state model with a generalized linear model (GLM) for the emissions distribution, typically using a Poisson process. In the case of the switching LDS, the generator includes a discrete variable that allows the model to switch between linear dynamics. GCLDS employs a generalized count distribution for the emissions distribution. Finally, in the case of PfLDS, a nonlinear feed-forward function (neural network) is inserted between the LDS and the GLM.

Gaussian process models have also been explored. GPFA[17] uses Gaussian processes (GPs) to infer a time constant with which to smooth neural data and has seen widespread use in experimental laboratories. More recently, the authors of [1] have used a variational approach (vLGP) to learn a GP that then passes through a nonlinear feed-forward function to extract the single-trial dynamics underlying neural spiking data.

Additional work applying variational auto-encoding ideas to recurrent networks can be found in [18]. The authors of [11] have defined a very general nonlinear variational sequential model, which they call the Deep Kalman Filter (DKF). The authors of [19] applied recurrent variational architectures to problems of control from raw images. Finally, [20] applied dynamical variational ideas to sequences of images. Due to the generality of the equations in many of these references, LFADS is likely one of many possible instantiations of a variational recurrent network applied to neural data (in the same sense that a convolutional network architecture applied to images is also a feed-forward network, for example).

The LFADS model decomposes the latent code into an initial condition and a set of innovation-like inferred inputs that are then combined via an RNN to generate dynamics that explain the observed data. Recasting our work in the language of Kalman filters, our nonlinear generator is analogous to the linear state estimator in a Kalman filter, and we can loosely think of the inferred inputs in LFADS as innovations in the Kalman filter language. However, an "LFADS innovation" is not strictly defined as an error between the measurement and the read-out of the state estimate. Rather, the LFADS innovation may depend on the observed data and the generation process in extremely complex ways.

Supplementary References

1. Zhao, Y. & Park, I. M. Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains. *Neural Comput.* **29,** 1293–1316 (2017).
2. Linderman, S. *et al.* Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. *Artificial Intelligence and Statistics* 914–922 (2017). at <http://proceedings.mlr.press/v54/linderman17a.html>
3. Sussillo, D. & Abbott, L. F. Generating coherent patterns of activity from chaotic neural networks. *Neuron* **63,** 544–557 (2009).
4. Mante, V., Sussillo, D., Shenoy, K. V & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503,** 78–84 (2013).
5. Carnevale, F., de Lafuente, V., Romo, R., Barak, O. & Parga, N. Dynamic Control of Response Criterion in Premotor Cortex during Perceptual Detection under Temporal Uncertainty. *Neuron* **86,** 1067–1077 (2015).
6. Sussillo, D., Churchland, M. M., Kaufman, M. T. & Shenoy, K. V. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat. Neurosci.* **18,** 1025–1033 (2015).
7. Rajan, K., Harvey, C. D. & Tank, D. W. Recurrent Network Models of Sequence Generation and Memory. *Neuron* **90,** 1–15 (2016).
8. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv [stat.ML]* (2013). at <http://arxiv.org/abs/1312.6114v10>
9. Rezende, D. J., Mohamed, S. & Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. in *International Conference on Machine Learning, 2014* (2014).
10. Chung, J. *et al.* A Recurrent Latent Variable Model for Sequential Data. in *Advances in Neural Information Processing Systems (NIPS)* (2015).
11. Krishnan, R. G., Shalit, U. & Sontag, D. Deep Kalman Filters. *arXiv Prepr. arXiv1511.05121* (2015).
12. Gregor, K., Danihelka, I., Graves, A., Rezende, D. J. & Wierstra, D. DRAW: A Recurrent Neural Network For Image Generation. *arXiv [cs.CV]* (2015). at <http://arxiv.org/abs/1502.04623>
13. Macke, J. H. *et al.* Empirical models of spiking in neural populations. *Advances in neural information processing systems* 1350–1358 (2011).
14. Petreska, B. *et al.* in *Advances in Neural Information Processing Systems 24* (eds. Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. & Weinberger, K. Q.) 756–764 (Curran Associates, Inc., 2011). at <http://papers.nips.cc/paper/4257-dynamical-segmentation-of-single-trials-from-population-neural-data.pdf>
15. Gao, Y., Buesing, L., Shenoy, K. V. & Cunningham, J. P. High-dimensional neural spike train analysis with generalized count linear dynamical systems. *Adv. Neural Inf. Process. Syst.* 1–9 (2015). at <https://bitbucket.org/mackelab/pop_spike_dyn/downloads/Gao_Buesing_2015_GCLDS.

pdf>

16. Gao, Y., Archer, E. W., Paninski, L. & Cunningham, J. P. in *Advances in Neural Information Processing Systems 29* (eds. Lee, D. D., Sugiyama, M., Luxburg, U. V, Guyon, I. & Garnett, R.) 163–171 (Curran Associates, Inc., 2016). at <http://papers.nips.cc/paper/6430-linear-dynamical-neural-population-models-through-nonlinear-embeddings.pdf>

17. Yu, B. M. *et al.* Gaussian-Process Factor Analysis for Low-Dimensional Single-Trial Analysis of Neural Population Activity. *J. Neurophysiol.* **102,** 614–635 (2009).

18. Bayer, J. & Osendorfer, C. Learning stochastic recurrent networks. *arXiv Prepr. arXiv1411.7610* (2014).

19. Watter, M., Springenberg, J., Boedecker, J. & Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. in *Advances in Neural Information Processing Systems* 2746–2754 (2015).

20. Karl, M., Soelch, M., Bayer, J. & van der Smagt, P. Deep variational Bayes filters: Unsupervised learning of state space models from raw data. *arXiv Prepr. arXiv1605.06432* (2016).