HHS Public Access

Author manuscript

Nat Methods. Author manuscript; available in PMC 2019 March 17.

Published in final edited form as:

Nat Methods. 2018 October ; 15(10): 805–815. doi:10.1038/s41592-018-0109-9.

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

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

[1]Wallace H. Coulter Department of Biomedical Engineering, Emory University and Georgia Institute of Technology, Department of Neurosurgery, Emory University, Atlanta, Georgia, USA

[2]Department of Neurosurgery, Stanford University, Stanford, California, USA

[3]Department of Electrical Engineering, Stanford University, Stanford, California, USA

[4]Stanford Neurosciences Institute, Stanford University, Stanford, California, USA

[5]Neurosciences Graduate Program, Stanford University, Stanford, California, USA

[6]Department of Neurobiology, Department of Bioengineering, Bio-X Program, Stanford University, Stanford, California, USA

[7]Google AI, Google Inc. Mountain View, California, USA

[8]Department of Electrical Engineering, University of California, Los Angeles, California, USA

[9]Cold Spring Harbor Laboratory, Cold Spring Harbor, New York, USA

[10]Department of Neurosurgery, Palo Alto Medical Foundation, Palo Alto, California, USA

[11]Center for Neurorestoration and Neurotechnology, Rehabilitation R&D Service, Department of VA Medical Center, Providence, Rhode Island, USA

[12]Center for Neurotechnology and Neurorecovery, Department of Neurology, Massachusetts General Hospital, Harvard Medical School, Boston, Massachusetts, USA

[13]School of Engineering, Brown Institute for Brain Science, Brown University, Providence, Rhode Island, USA

*correspondence should be addressed to David Sussillo sussillo@google.com and Chethan Pandarinath chethan@gatech.edu.

^work done while at.

Author Contributions

C.P., D.J.O., and D.S. designed the study, performed analyses, and wrote the manuscript with input from all authors. D.S. and L.F.A. developed the algorithmic approach. C.P., J.C., and R.J. contributed to algorithmic development and analysis of synthetic data. D.J.O., S.D.S., J.C.K., E.M.T., M.T.K., S.I.R., and K.V.S. performed research with monkeys. C.P., L.R.H., K.V.S., and J.M.H. performed research with human research participants. All authors contributed to revising the manuscript.

Data Availability

Data will be made available upon reasonable request from the authors, unless prohibited due to research participant privacy concerns.

[14]Zuckerman Mind Brain Behavior Institute, Department of Neuroscience, Department of Physiology and Cellular Biophysics, Columbia University, New York, New York, USA

[15]Howard Hughes Medical Institute, Stanford University, Stanford, CA, USA

Abstract

Neuroscience is experiencing a revolution in which simultaneous recording of many thousands of neurons is revealing population dynamics that are not apparent from single-neuron responses. This structure is typically extracted from trial-averaged data, but deeper understanding requires studying single-trial phenomena, which is challenging due to incomplete sampling of the neural population, trial-to-trial variability, and fluctuations in action potential timing. We introduce Latent Factor Analysis via Dynamical Systems (LFADS), a deep learning method to infer latent dynamics from single-trial neural spiking data. LFADS uses a nonlinear dynamical system to infer the dynamics underlying observed spiking activity and to extract 'de-noised' single-trial firing rates. When applied to a variety of monkey and human motor cortical datasets, LFADS predicts observed behavioral variables with unprecedented accuracy, extracts precise estimates of neural dynamics on single trials, infers perturbations to those dynamics that correlate with behavioral choices, and combines data from non-overlapping recording sessions spanning months to improve inference of underlying dynamics.

Introduction

Increasing evidence suggests that in many brain areas, the activity of large populations of neurons is often well-described by low-dimensional dynamics (e.g. [1–9]). These findings suggest that one can begin to understand the computations of brain areas without observing all their neurons because these computations can be described by the time-varying activity and interactions (i.e., dynamics) of a modest number of underlying 'latent factors'[10]. Recovering these dynamics on single trials is essential for illuminating the relationship between neural population activity and behavior, and for advancing therapeutic neurotechnologies such as closed-loop deep brain stimulation and brain-machine interfaces. However, recovering population dynamics on single trials is difficult due to trial-to-trial variability (e.g. behavior or arousal state) and fluctuations in the spiking of individual neurons. Standard analyses sacrifice single-trial information for the sake of better estimates of trial-averaged neural states[3,6,7,11]. Current techniques for extracting neural population states from single trials typically make simplifying assumptions by modeling the underlying population dynamics as having independent underlying factors[12,13], as being linear[14–17] or as being switched linear[18,19].

Here we introduce a novel machine learning method based on nonlinear artificial recurrent neural networks (RNNs), termed Latent Factor Analysis via Dynamical Systems (LFADS, "*ell-fads*"). LFADS is based on the simple conceptual idea that neural data can be generated by a dynamical system. LFADS models the following generic dynamical system,

$$\dot{\mathbf{x}}(t) = \mathrm{F}\left(\mathbf{x}(t), \mathbf{u}(t)\right). \quad (1)$$

The state of the dynamical system **x**(t) is updated by the vector-valued function **F**(), which is nonlinear and potentially complicated, accepts optional input **u**(t), and is seeded by an initial condition, **x**(0). LFADS models **F**(), **x**(0), and optionally **u**(t). By modeling equation (1) LFADS assumes that the underlying process that produces the observed spiking activity can be modeled as a dynamical system. The input, if it is used, is constrained to be considerably less dynamically complex than **x**(t). Without such a condition, equation (1) does not constrain the data.

The applicability of equation (1) to neural data relies on four assumptions, namely, that spiking activity on a single trial of a task depends on: 1) underlying dynamics (i.e., rules by which neural activity evolves in time) that govern the brain area(s) being recorded; 2) trial-specific initial conditions that reflect the state of the neural population at a specific point in time; 3) effects of unmeasured inputs from other brain areas, including those arising from unexpected changes in the task, contextual inputs, or sensory inputs, and 4) Poisson spiking variability.

We now move towards a concrete implementation that can take observed neural data as an input and provide estimates of these data's governing neural dynamics, specifically, its latent neural state, initial conditions, inputs, and de-noised firing rates (rates). In LFADS, the underlying dynamics (assumption 1) are generated by an RNN (the "generator"). A core assumption is that the dynamics of neural data generated by a biological network can be described by a continuous valued dynamical system. Dynamic "factors" are extracted from this system (an RNN) and used to generate (and thereby infer) rates for the recorded neurons. Observed action potentials are modeled as samples from an inhomogenous Poisson process whose rate corresponds to the inferred firing rate for the given neuron (assumption 4). Initial conditions and input for the generator (assumptions 2 and 3) are extracted from the observed spiking data for each trial by additional RNNs (the "encoder" and "controller"). Yet, beyond binned spike sequences, no other trial-specific information is supplied (i.e., no condition or behavioral information).

The strength of this approach lies in the ability of nonlinear RNNs to reproduce the complex temporal activity patterns that underlie the neural data. In addition, LFADS can find low-dimensional dynamics that explain the recorded data because the number of factors in the model may be deliberately constrained. This is consistent with repeated empirical observations that the dimensionality of neural population activity in areas like motor and prefrontal cortices is, in many cases, much lower than the number of recorded neurons[3,7,20,21] (discussed in [22]).

Here we apply LFADS to a variety of datasets from rhesus macaque motor (M1) and pre-motor (PMd) cortices, as well as human M1 (datasets are outlined in Online Methods Table 2; macaque data were previously recorded at Stanford University). We show that rates extracted by LFADS can be used to estimate behavioral variables (e.g., reaching kinematics) significantly more accurately than other techniques. We also show in single trials that the dynamics inferred by LFADS capture previously-uncovered rotational dynamics found in condition-averaged data, and that the learned dynamical system is predictive of behavioral conditions (e.g., reach types) that it was not trained to model. Further, we demonstrate that

LFADS can combine data from non-overlapping recording sessions, each sampling from separate neural populations and spanning 5 months of recording, to improve its performance on the individual trials from each recording session. Finally, we demonstrate the ability of LFADS to infer inputs to a neural circuit by analyzing data from an arm-reaching task involving a mid-trial perturbation, and by testing whether it can uncover high-frequency oscillations in the underlying rates associated with local field potentials.

Results

Overview of LFADS

To begin (Figs. 1–4 and Supp. Figs. 1–6), we use a simplified conceptual dynamical systems model that ignores the input in equation (1), yielding

$$\dot{\mathbf{x}}(t) = \mathrm{F}\left(\mathbf{x}(t)\right). \quad (2)$$

This means that beyond Poisson spiking variability, all trial-to-trial variability is captured by the initial condition, $\mathbf{x}(0)$, for that trial. We transition to concrete language to properly describe the LFADS architecture.

LFADS is a sequential adaptation of a variational auto-encoder[23,24] constructed by maximizing a lower bound on the likelihood of the observed spiking activity given the rates produced by the generator network, across all model training trials. Parameters are learned using backpropagation (full model details and training procedures are given in [25] and Online Methods, and associated source code is available).

Working from output (right) to input (left) in (Fig. 1a), LFADS models the single-trial spiking observations at time t as stochastic (Poisson) spike counts generated from a vector of underlying firing rates $\boldsymbol{r_t}$. For neuron i, the LFADS-inferred rate $r_{t,i}$ provides a de-noised rate for its observed spiking activity on a trial-by-trial basis. The rates are obtained by multiplying a vector of dynamic factors $\boldsymbol{f_t}$ by a readout matrix $\boldsymbol{W^{rate}}$ and exponentiating the resulting quantity. These factors are determined by multiplying the vector of activities $\boldsymbol{g_t}$ of the generator by a matrix $\boldsymbol{W^{fac}}$. The activities of the generator's units depend on two elements: a trial-specific initial state vector $\boldsymbol{g0}$ (one for each trial), and the parameters defining the connections of the network (fixed across trials after training). The units of the generator are not meant to correspond directly to any recorded channels, but rather, the generator is meant to model the dynamics underlying the observed data. The inferred initial state $\boldsymbol{g0}$ is provided by a linear readout of the activity of the encoder. To compute $\boldsymbol{g0}$ for a given trial, the encoder receives a temporal sequence of the vectors of recorded (binned) spike counts for that trial. To better model the trials, the encoder runs through the trial both backwards and forwards to compute $\boldsymbol{g0}$, meaning that when generating the trial at any time t, LFADS has access to data before and after t.

Once the model has been trained, spike counts from a specific trial are fed into the encoder, which infers initial conditions for that trial (Fig. 1a). The encoder compresses the temporal sequence of spiking data for each trial into a single vector - the "latent code" - which is the

initial condition to the generator. From this compressed code, the generator infers the factors and rates of all the recorded neurons across time for the encoded trial (in Supp. Fig. 1 we apply LFADS to a 1-D pendulum to show how LFADS operates for a simple dynamical system). Thus, LFADS turns time series of single-trial recorded spike counts into low-dimensional dynamic factors and underlying rates which generated the observed spikes.

We begin by training LFADS on multielectrode array data (single-trial spiking activity) from M1 and PMd, recorded while a monkey made reaching movements (for model training details, see Online methods, and Online Methods Table 1 for all model hyperparameters). The analyzed trials were 800 ms long and aligned to movement onset (i.e., the time when arm movement was first detectable). Inferred rates and factors for seven example trials are shown (Fig. 1b).

We first assessed the validity and accuracy of rates and factors inferred by LFADS from simulated data for which the ground-truth is known (summarized in Online Methods, section 2.1). These simulations show LFADS outperforms a number of state-of-the-art machine-learning techniques (GPFA[12]; PfLDS[15]; and vLGP[13]). Assessing the quality and validity of results on real data (e.g., Fig. 1b) is difficult because the ground-truth is either unknown or non-existent (e.g., because single-trial "instantaneous" firing rates are abstractions rather than experimentally measurable phenomena). Thus, by “validation”, our intent is to demonstrate that applying LFADS, i.e., applying a nonlinear dynamical systems model to the data, provides an informative description of the observed data by leading to superior correlation with behavior such as kinematics, or reproducing on single trials phenomena previously reported using condition-averaging.

We next tested the validity of LFADS-inferred factors and rates by verifying that they: reproduce features seen in common neuroscientific analyses (PSTHs, cross-correlations; Fig. 2, Supp. Data 1–3); are predictive of held-out, simultaneously-recorded neurons (Fig. 2); predict details of behavior (Figs. 2, 4, 5, Supp. Fig. 6); exhibit single-trial features previously demonstrated in trial-averaged analysis (Fig. 3); predict held-out conditions (Fig. 3); and correlate with local field potentials (LFPs) (Fig. 6).

Validation of LFADS inferences using a complex reaching task

We applied LFADS to 202 neurons simultaneously recorded from M1/PMd during a “Maze” task (see Online Methods) in which a monkey made a variety of straight and curved reaches (Fig. 2a; dataset consisted of ~2300 individual reach trials spanning 108 reach types). In all examples we show, LFADS was trained to model observed spiking data from individual trials *without* any information about task conditions or behavioral parameters (e.g. reach kinematics or EMG).

We first compared LFADS-inferred rates to smoothed spikes and to Gaussian Process Factor Analysis (GPFA[12])-inferred rates (Fig. 2b). Condition-averaged smoothed spikes are commonly known as the peri-stimulus time histogram (PSTH); these assume that rates are smooth in time and consistent across repetitions of an individual condition, while GPFA assumes that population activity is low dimensional and smooth in time on several characteristic timescales. Finally, LFADS assumes that rates are predictable, i.e., they evolve

from an initial condition of a dynamical system, and also potentially low-dimensional. These differing assumptions lead to different condition-averaged and single-trial rates. Qualitatively, the condition-averaged LFADS-inferred rates were similar to the PSTHs calculated from the observed spiking data. We also compared single-trial LFADS-inferred rates to single-trial rates constructed by smoothing spikes or using GPFA. The single-trial LFADS-inferred rates show far more structure than those from smoothing spikes or GPFA. When compared to GPFA, the LFADS-inferred rates preserved many of the faster timescale features of the neurons' PSTHs (4 neurons shown; all PSTHs are included as Supp. Data 1). Finally, we showed that LFADS-inferred rates reproduce patterns of correlations across time (Supp. Data 2) and neurons (Supp. Data 3) for different behavioral conditions. As with the PSTHs, the cross-correlograms inferred by LFADS reproduced the structure of the empirical cross-correlograms, and particularly preserved faster timescale temporal features better than GPFA.

LFADS encodes each individual trial by an initial state vector (***g0***). To test whether there was behaviorally-relevant structure in the ***g0*** encoding, we applied a widely used nonlinear dimensionality reduction technique, t-distributed stochastic neighbor embedding (t-SNE; Fig. 2c). After using t-SNE to reduce the dimensionality to 3, we color coded the ~2300 points based on the angle of the target of the upcoming reach. As shown, t-SNE uncovered clear structure in the learned ***g0*** encoding, specifically, trials with similar kinematic structure are encoded with similar initial conditions (further detail, e.g. separation between curved vs. straight reaches, can be seen in Supp. Video 1). Critically, this demonstrates that the generator is not learning arbitrary sequences to model each trial, but instead learning an organized representation that preserves the relation of the trials in kinematic space.

We also tested whether the LFADS-inferred representations were informative about behavioral parameters, specifically, the trajectory of the monkey's hand movements (Fig. 2d). Hand velocities were estimated from LFADS-inferred rates using cross-validated optimal linear estimation (OLE[26]). Using the full population of 202 neurons, decoding using LFADS-inferred rates dramatically outperformed results obtained by binning or smoothing spike trains, or by using GPFA (average R^2 of 0.90 across the dataset, vs. 0.66, 0.69, and 0.34 for smoothing, GPFA, and binning, respectively. Note: for offline analysis, the smoothing approach is a generalization of common brain-machine interface decoders such as the Kalman filter; detailed in [27] and Online Methods). We also determined performance as a function of population size by drawing random sub-samples from the neural population (Fig. 2e). LFADS using 25 (X velocity) or 50 (Y velocity) neurons outperformed the other techniques applied to the full population of 202 neurons. The bin size and number of factors used by LFADS (5 ms and 20, respectively) were held constant for all models across all population sizes, while the bin size and number of factors used for GPFA were chosen to optimize decoding accuracy.

We also tested whether the LFADS-inferred low-dimensional factors were predictive of held-out data (Fig. 2f). Because the factors reflect the full neural population dynamics, they should be predictive for neurons that were not used to train the model (i.e., held-out neurons). We fit LFADS models to subsets of neurons (25, 50, 100, and 150 neurons were drawn from the full population of 202 neurons). We then used a standard Generalized Linear

Model (GLM) to relate the LFADS-inferred factors to the held-out neurons' spike counts in a cross-validated manner. For each held-out neuron, a GLM was trained to relate inferred factors to observed spike counts for a training subset of trials, and rates were predicted for that neuron for a test subset using the trained GLM. The rates produced by LFADS-inferred factors were predictive of spiking activity for held-out neurons on held-out trials, providing improved single-trial likelihood over the factors inferred by GPFA ($p<10^{-8}$ for all population sizes, Wilcoxon signed-rank test).

Uncovering rotational dynamics in motor cortex

We next tested whether the population dynamics inferred by LFADS on single trials exhibited dynamic features that have previously been identified by analyzing trial-averaged data, specifically, the rotational dynamics underlying M1/PMd firing rates that accompany the transition from pre- to peri-movement activity in monkeys[3] and humans[8]. Rotational dynamics were consistent across the full range of movements being performed (Fig. 3a, monkey J, 108 reach conditions of the maze dataset, and Fig. 3c, participant T5, 8 attempted movement conditions in a "center-out" task). These results were obtained by averaging the rate of each neuron across all trials corresponding to a particular reach condition (condition-averaging), and then applying a form of dimensionality reduction (jPCA[3]). Although condition-averaging reveals the basic oscillatory dynamics, single trials provide noisy and unstructured views of the neural trajectories (Figs. 3b & 3d). In contrast, applying jPCA to the LFADS-inferred rates shows that LFADS not only reproduces the previously-extracted oscillatory dynamics on a condition-averaged basis (Figs. 3e & 3g), it also demonstrates, for the first time, the presence of rotational dynamics on single trials (Fig. 3f, Supp. Video 2, monkey J, 2296 maze reaching trials, and Fig. 3h, participant T5, 114 center-out movement attempts).

We next tested whether the LFADS generator learns dynamics that generalize to new conditions (Fig. 3i-k). If the dynamical systems model of M1 is appropriate, then after learning the population's underlying dynamics, it should be possible to generate activity from any novel, unseen reaching condition simply by knowing the proper initial state. After setting the initial state, the learned dynamics model should then generate the appropriate time-varying activity for the novel condition. To test whether this is the case, data were split into training conditions and held-out (validation) conditions based on target angle (Fig. 3i). (Briefly, the workspace was uniformly divided into angular bins, and conditions were grouped by the position of their reach target. This resulted in 19 sets of conditions; see Online Methods.) For each set, an LFADS model was trained solely on the 18 other training condition sets, and then evaluated on the held-out set. We then collated LFADS-inferred rates for all the held-out trials (combining data from 19 LFADS models - one model per held-out condition set), and projected them into the jPCA plane previously found using all data (Fig. 3j). As shown, even though the generator had not been trained on the held-out trials, it still modeled them with rotational dynamics, in the same plane as found previously. Finally, we compared the initial position in the jPCA plane found when a trial is held-in, vs. held-out, and found a clear correlation (Fig. 3k). This proof-of-principle analysis demonstrates that LFADS can learn dynamics that generalize to completely novel

conditions, provided datasets with sufficient trial counts and diverse conditions to capture the neural population's dynamics.

Stitching together data from multiple sessions

Thus far, we have demonstrated the application of LFADS to data recorded from single neural populations. However, experiments are often performed across multiple sessions, with different neurons recorded on each session (e.g., using acute probes that are placed independently each session). LFADS provides a new ability to "stitch" such data together to create a more powerful and comprehensive dynamical model. The aim is similar to previous efforts to relate separately recorded neural population activity[28,29], but importantly, LFADS relates the separate sessions through a learned nonlinear dynamical system, and does not require any overlap between the populations of recorded neurons.

In experiments where a subject is engaged in the same behavior across recording sessions and the same brain region is being recorded, a reasonable hypothesis is that separately recorded neural populations participate in the same underlying dynamics. LFADS is well-suited to leverage this structure because of its two-step process of inference (Fig. 4a). To stitch multiple sessions into a common dynamical model, we configure LFADS to use per-session "read-in" matrices $\boldsymbol{W^{input}}$, mapping from observed spiking to input factors, and "read-out" matrices $\boldsymbol{W^{rate}}$, mapping from factors to neuron rates. The shape of these matrices can vary to match the number of neural channels recorded in each dataset. Importantly, a single encoder, generator, and factor matrix $\boldsymbol{W^{fac}}$ are shared across sessions and learned from all sessions. The per-session read-in and read-out matrices are learned using data from only the corresponding session (or precomputed; see Online Methods).

We tested this approach using neural activity from monkey M1 and PMd during a center-out instructed-delay reaching task, recorded using linear multielectrode arrays (monkey P; 24 channel V-probes, Plexon). We trained one stitched multi-session LFADS model on a combined dataset consisting of 44 recording sessions that spanned 162 days (Fig. 4b shows locations of the 38 individual penetration sites in the precentral gyrus, and Fig. 4c shows sample recordings from 6 sessions). We then examined the condition-averaged factor trajectories inferred for each recording session. These trajectories are highly similar for a given reach direction regardless of the recording session (Fig. 4d), a key indication that LFADS found a generator capable of describing all datasets with a consistent set of factors. Single-trial factor trajectories also exhibited consistency across recording sessions (Fig. 4g, Supp. Fig. 5, Supp. Video 3).

We then compared the multi-session stitched LFADS model to 44 models trained using data from individual sessions. This comparison tests whether access to multiple M1 recordings allows multi-session LFADS to better model the underlying population dynamics. We assessed the quality of the LFADS models by asking how informative the factors ($\boldsymbol{f_t}$) were in predicting behavioral observations, including reach kinematics and reaction times. In this case we decoded from the factors because, for the multi-session model, they are common across all recording sessions and therefore are enriched by the additional sessions. Consistent with previous analyses, the single-session LFADS models produced factors that were substantially more predictive of kinematics than Gaussian-smoothed spiking (mean

improvement of 0.32 in R^2; $p < 10^{-8}$, Wilcoxon signed-rank test) or GPFA (mean improvement of 0.27 in R^2, $p < 10^{-8}$, Wilcoxon signed-rank test; Fig. 4e), indicating that LFADS identified useful dynamic representations even from the limited observations from individual recording sessions. Importantly, however, the stitched LFADS model produced factors that were considerably more informative than the single-session LFADS models, resulting in significantly improved kinematic predictions, even when using a single decoder across all sessions (mean increase of 0.22 in R^2, $p < 10^{-8}$, Wilcoxon signed-rank test; Fig. 4e,f). We note that the lower decoding fidelity in the current experiment, in comparison to Fig. 2, likely arises from the difference in recording methodologies: the dataset from Fig. 2 consisted of 202 neurons recorded using two 96-channel Utah arrays (192 total channels). We also predicted reaction time from LFADS factors (Supp. Fig. 6); again, the stitched model significantly outperformed the single-day models (mean improvement in correlation coefficient between predicted and measured reaction times: 0.15; $p < 10^{-7}$, Wilcoxon signed-rank test).

Inferring inputs to a neural circuit

We next adapt LFADS to model the more general dynamical system of equation (1), i.e., we introduce inputs to allow the neural population activity to be modeled as a non-autonomous dynamical system. This capacity is critical when a neural population is driven by unmeasured inputs from other brain areas, including those arising from unexpected changes in the task, contextual inputs, or sensory inputs. Conceptually, inferring the presence of inputs requires building an accurate model of the observed population's internal dynamics. With such a model, it should be possible to determine when data deviate from the model's dynamic predictions. This indicates that an external perturbation to the system occurred, which can be captured as an *inferred* input - inferred because LFADS models the input which supplies the deviation from the unperturbed dynamics (we outline caveats in the Discussion). The remainder of the results (Figs. 5–6, Supp. Figs. 7–9) use this more general LFADS model. For these examples, this means that beyond Poisson spiking, trial-to-trial variability is captured by both the initial condition $\boldsymbol{g_0}$ and the inferred input $\boldsymbol{u_t}$ for that trial.

To test LFADS's ability to infer inputs, we analyzed data from a "Cursor Jump" task in which a monkey guided a cursor, controlled by the monkey's hand position, towards upward or downward targets (monkey J; see Online Methods). The target position was shown to the monkey starting at the beginning of the trial. On "unperturbed" trials (75%), the cursor consistently tracked the position of the monkey's hand, and the monkey made straight upward or downward reaching movements to acquire targets. On "perturbed" trials (25%), unpredictable shifts to the left or right between cursor and hand position forced the monkey to make corrective movements to acquire the target (Fig. 5b). We applied LFADS to spiking activity from multielectrode arrays implanted in M1/PMd (Fig. 5c), allowing four inferred inputs (choice of dimensionality detailed in Online Methods). We analyzed the first 800 ms of each trial, beginning at target onset (jumps occurred ~350–550 ms later).

LFADS used inferred inputs to model information flow into the generator with timing that was consistent with the trial structure. Prior to the trial, the monkey had no information about the target position, which was cued at the beginning of the trial (target onset). Around

this time, the inferred inputs are distinct with respect to target position (Fig. 5d, e.g. Input dim 1, comparing inputs inferred for Upward vs. Downward trials), but are not distinct with respect to perturbation type (i.e., red, blue, and grey traces are overlapping), as perturbations occurred much later in the trial. In contrast, around the time of perturbation, LFADS inferred different input patterns for right- and left-shift perturbed trials and for unperturbed trials (Fig. 5d, red, blue, and grey traces, e.g. Input dim 2). Furthermore, the timing of these inputs is well-aligned to the time of the perturbations (which were variable), and the perturbation direction specificity of these inputs were similar across downward and upward reaches (Fig. 5d, top and bottom panels). The trends were also visible on single-trials (Supp. Fig. 10). We applied t-SNE to the inferred single-trial inputs around the time of the perturbation (Fig. 5e), which revealed that they cluster according to perturbation identity on a single-trial basis. We note that the exact shape of the inferred inputs may not resemble physiological signals. In addition, because the LFADS encoding is acausal, the timing of the inputs is not required to be causal relative to the timing of the perturbations (see Discussion). Nevertheless, this example demonstrates the ability of LFADS to predict, on average, the presence, identity, and timing of inputs to motor cortex related to task perturbations.

LFADS rate oscillations correlate with local field potentials

Another known dynamic feature of motor cortical activity is the rhythmic spiking that often occurs during the pre-movement period, typically phase-locked to accompanying LFP oscillations (15–40 Hz; e.g., [30,31]). We tested whether LFADS is capable of extracting such high-frequency dynamic features. Previous work has hypothesized that spike-LFP phase locking is reflective of communication between brain areas[32]. Therefore, we reasoned that inputs were necessary to model these high-frequency oscillations. Indeed, when LFADS was allowed to use inputs, high-frequency oscillations were evident in the inferred rates (Fig. 6a). Although the model was not given access to the LFPs, the inferred oscillations aligned well with LFPs and with structure apparent in the multi-unit spiking activity (Fig. 6a).

We studied the spike-LFP phase locking in monkey and human data using cross-correlation analysis (Fig. 6b, black traces). Cross-correlations were computed on a single-trial basis, using data from the first 250 ms (monkey) or 300 ms (human) of each trial, and then averaged over trials. As shown, this is a single-trial phenomenon: high-frequency oscillations in the cross-correlograms disappear when they are computed after shuffling trial identity (Fig. 6b, blue traces).

We also studied the correlation between the LFADS-inferred rates and the LFP on single trials, which was strikingly similar to the spike-LFP phase locking (Fig. 6b, red traces), confirming LFADS's ability to uncover high-frequency dynamic features. We note that we were unable to robustly reproduce the correlations between LFADS-inferred rates and LFP on held out trials without the use of inferred inputs. This suggests that these fast dynamics are not dynamical in the sense of being able to be generated with an autonomous dynamical system using only an initial condition to describe trial-to-trial variability.

Discussion

The increasing ability to record from large ensembles of neurons has inspired a shift from emphasizing the properties of individual neurons and their responses to exploring the emergent properties of neural populations. Such efforts reinforce theoretical work that suggests that emergent dynamics may serve as one of the brain's fundamental computational mechanisms (reviewed in [33]). LFADS provides a novel approach toward building empirical models of the dynamics underlying population activity, and leverages these dynamics models to infer latent representations that are considerably more informative about subjects' behaviors than the observed population activity itself. The close link between the LFADS-inferred representations and subjects' behaviors, especially on a single-trial, moment-by-moment basis, lends strong evidence to suggest that network states and dynamics, rather than the properties of individual neurons, are a key factor in understanding the computations performed by brain areas and how they ultimately mediate behaviors.

How seriously should the structure of the LFADS generator be taken as a model of a brain region one is studying? More theoretical work is required to answer this question. Artificial RNNs and biological RNNs provide different substrates for implementing computation through dynamics, and the LFADS architecture does not resemble the biophysical architecture of the cortex. Of course, if one desires to study biophysical detail, then it is critical to build a biophysically detailed network model. We advise against making inferences about properties of the biological network by studying the structure of the generator. Instead, we believe that LFADS can identify abstract dynamics that approximate the progression of neural state changes related to spiking, without modeling the specific biological components, ultimately producing an abstract model that captures the computations being performed by the network under study.

LFADS also provides a new avenue toward distinguishing the dynamics internal to a neural circuit from the influence of unmeasured input from other brain regions, which is a vexing challenge in neuroscience. Although the nature of the inputs inferred by LFADS is informative about the presence and identity of perturbations, caution should be used when interpreting the precise shape and timing of these inputs. In addition to reflecting actual inputs to a neural ensemble, LFADS-inferred inputs may capture model mismatch (e.g. biophysical spiking vs. Poisson process) and measurement noise. There is no constraint requiring the inferred inputs' shapes to conform to physiological processes. Furthermore, their timing may be imprecise relative to the timing of the perturbations they describe. Finally, due to the bidirectional encoders used by LFADS, the generator has access to the entire data sequence. There is no constraint forcing the inputs to be causal with respect to the task perturbation. Caveats aside, both the presence, timing, and qualitative shape of the inferred input in the Cursor Jump task (also two synthetic examples) are reasonable, providing evidence that inputs inferred by LFADS are useful for thinking about neural computations by disambiguating internal dynamics from input driven dynamics.

A guiding factor in choosing model hyperparameters is constraining the complexity of the reduced-dimensional representations. This is especially critical in the case of the inferred inputs, which provide a potential method for the system to forego modeling dynamics

altogether when reconstructing the data. In a limit case, one could match the number of inferred inputs to the dimension of the observed data, allowing a potential identity mapping that would produce inputs that essentially replicate the observed spike times (a concern common to all auto-encoders). Such a model might produce an accurate reconstruction of the observed data without "learning" anything useful. Due to this confound, reconstruction cost is not an ideal metric for evaluating the performance of a model in inferring the population's dynamics, and future work must address this challenge. At present, a reasonable approach is to use as few inferred inputs as possible to force the LFADS generator to model the population's underlying dynamics.

LFADS provides several capabilities that will be critical to understanding the role of computation through dynamics in many brain areas that have previously been difficult to study. For example, modeling a population's *internal* dynamics may be crucial in studying neural computations that have no clear, observable *external* behavioral correlates on a moment-by-moment basis, such as integration of evidence during decision-making tasks, or attentional regulation. Additionally, a causal variant of LFADS could improve performance of therapeutic neurotechnologies that rely on real-time neural state estimation, such as brain-machine interfaces (BMIs), which decode movement intention in real-time to control external devices[34–37], or closed-loop neuromodulation approaches, which require real-time neural state estimates to guide stimulation[38–41]. In addition, the ability of stitching to improve neural state estimates by combining multiple recordings may improve stability of these devices. Taken together, the capabilities produced by LFADS have the potential to yield powerful new approaches to understand neural computation and dynamics in many new frontiers, and to apply this knowledge towards the treatment of diverse neurological disorders.

Online Methods

1 The LFADS Model

1.1 Code availability

- Source code for LFADS can be found at https://github.com/tensorflow/models/tree/master/research/lfads.
- Source code for interfacing LFADS with MATLAB can be found at https://github.com/lfads/lfads-run-manager.
- Extensive technical documentation for the source code can be found at https://lfads.github.io/lfads-run-manager.

1.2 The variational auto-encoder—The LFADS model is an instantiation of a variational auto-encoder (VAE)[23,43] extended to sequences, as in [44] or [45]. The VAE consists of two components, a decoder (also called a generator) and an encoder. The generator assumes that data, denoted by $\mathbf{x}$, arise from a random process that depends on a vector of stochastic latent variables $\mathbf{z}$, samples of which are drawn from a prior distribution $P(\mathbf{z})$. Simulated data points are then drawn from a conditional probability distribution, $P(\mathbf{x}|\mathbf{z})$ (we

have suppressed notation reflecting the dependence on parameters of this and the other distributions we discuss).

The VAE encoder transforms actual data vectors, $\mathbf{x}$, into a conditional distribution over $\mathbf{z}$, $Q(\mathbf{z}|\mathbf{x})$. $Q(\mathbf{z}|\mathbf{x})$ is a trainable approximation of the posterior distribution of the generator, $Q(\mathbf{z}|\mathbf{x}) \approx P(\mathbf{z}|\mathbf{x}) = P(\mathbf{x}|\mathbf{z})P(\mathbf{z})/P(\mathbf{x})$. $Q(\mathbf{z}|\mathbf{x})$ can also be thought of as an encoder from the data to a data-specific latent code $\mathbf{z}$, which can be decoded using the generator (decoder). Hence the auto-encoder; the encoder Q maps the actual data to a latent stochastic “code", and the decoder P maps the latent code back to an approximation of the data. Specifically, when the two parts of the VAE are combined, a particular data point is selected and an associated latent code, $\hat{\mathbf{z}}$ (we use $\hat{\mathbf{z}}$ to denote a sample of the stochastic variable $\mathbf{z}$) is drawn from $Q(\mathbf{z}|\mathbf{x})$. A data sample is then drawn from $P(\mathbf{x}|\hat{\mathbf{z}})$, on the basis of the sampled latent variable. If the VAE has been constructed properly, $\hat{\mathbf{x}}$ should resemble the original data point $\mathbf{x}$.

The loss function that is minimized to construct the VAE involves minimizing the Kullback-Leibler divergence between the encoding distribution $Q(\mathbf{z}|\mathbf{x})$ and the prior distribution of the generator, $P(\mathbf{z})$, over all data points. In the VAE framework $P(\mathbf{z})$ is typically defined as a Gaussian prior whose parameters are independent of the data. The rationale is that even a simple distribution, such as a Gaussian, can be transformed into a complex distribution by passing samples of the Gaussian distribution through a powerful nonlinear function. One optimizes the parameters in order to maximize the likelihood of the data while reducing the distance between $Q(\mathbf{z}|\mathbf{x})$ and $P(\mathbf{z}|\mathbf{x})$. In the end, statistically accurate generative samples of the data can be created by running the generator model seeded with samples from $P(\mathbf{z})$, i.e. accurate samples of the data can be generated from white noise.

We now translate this general description of the VAE into the specific LFADS implementation aimed at high-dimensional, simultaneously recorded neural spike trains. Borrowing some notation from (Gregor et al., 2015), we denote an affine transformation ($\mathbf{v} = \mathbf{W}\,\mathbf{u} + \mathbf{b}$) from a vector-valued variable $\mathbf{u}$ to a vector-valued variable $\mathbf{v}$ as $\mathbf{v} = \mathbf{W}(\mathbf{u})$, we use $[\cdot,\cdot]$ to represent vector concatenation, and we denote a temporal update of a recurrent neural network receiving an input as $state_t = RNN^a(state_{t-1}, input_t)$, for an RNN named ’a’. It is understood that if there are two networks modules, such as RNNs, with different names, e.g. $RNN^a(.,.)$ and $RNN^b(.,.)$, these network modules do not share parameters.

1.3 LFADS Generator—The neural data we consider, $\mathbf{x}_{1:T}$, consists of spike trains from D recorded neurons. Our reference implementation of LFADS also supports continuous Gaussian distributed data, but as this is not central to the main application, we focus exclusively on spike trains in what follows. Each instance of a vector $\mathbf{x}_{1:T}$ is referred to as a trial, and trials may be grouped by experimental conditions, such as stimulus or response types. The data may also include an additional set of observed variables, $\mathbf{a}_{1:T}$, that may refer to stimuli being presented or other experimental features of relevance, such as kinematics. Unlike $\mathbf{x}_{1:T}$, the data described by $\mathbf{a}_{1:T}$ is not itself being modeled, but it may provide important conditioning information relevant to the modeling of $\mathbf{x}_{1:T}$. This introduces a slight complication: we must distinguish between the complete data set, $\{\mathbf{x}_{1:T}, \mathbf{a}_{1:T}\}$ and the part of the data set being modeled, $\mathbf{x}_{1:T}$. The conditional distribution of the generator, $P(\mathbf{x}|\mathbf{z})$, is only

over $\mathbf{x}$, whereas the approximate posterior distribution, $Q(\mathbf{z}|\mathbf{x},\mathbf{a})$, depends on both types of data.

LFADS assumes that the observed spikes described by $\mathbf{x}_{1:T}$ are samples from a Poisson process with underlying rates $\mathbf{r}_{1:T}$. Based on the dynamical systems hypothesis outlined in the introduction of the main text, the goal of LFADS is to infer a reduced set of latent dynamic variables, $\mathbf{f}_{1:T}$, of dimension F, from which the firing rates can be constructed. The rates are determined from the factors by an affine transformation followed by an exponential nonlinearity, $\mathbf{r}_{1:T} = \exp(\mathbf{W}^{rate}(\mathbf{f}_{1:T}))$. Note that $\exp(\cdot)$ is the inverse canonical link function for the Poisson distribution, making it a natural choice to keep the Poisson rate variable positive. The choice of a low-d representation for the factors is based on the observation that the intrinsic dimensionality of neural recordings tends to be far lower than the number of neurons recorded, e.g. [3,7,20], and see [22] for a more complete discussion.

The factors are generated by a recurrent nonlinear neural network and are characterized by an affine transformation of its state vector, $\mathbf{f}_{1:T} = \mathbf{W}^{fac}(\mathbf{g}_{1:T})$, with $\mathbf{g}_t$ of dimension N. Running the network requires an initial condition $\mathbf{g}_0$, which is drawn from a prior distribution $P^{g_0}\left(\mathbf{g}_0\right)$. Thus, $\mathbf{g}_0$ is an element of the set of the stochastic latent variables $\mathbf{z}$ discussed above.

There are different options for sources of time-dependent input to the recurrent generator network. First, as in some of the examples to follow, the network may receive no input at all. Second, it may receive the information contained in the non-modeled part of the data, $\mathbf{a}_{1:T}$, in the form of a network input. Instead, as a third option, we introduce an inferred input $\mathbf{u}_{1:T}$. When an inferred input is included, the set of stochastic latent variables is expanded to include it, $\mathbf{z} = \{\mathbf{g}_0, \mathbf{u}_{1:T}\}$. At each time step, $\mathbf{u}_t$ is drawn from a prior distribution $P^u(\mathbf{u}_t|\mathbf{u}_{t-1})$ that is auto-regressive, with $P^{u_1}\left(\mathbf{u}_1\right)$ defining the distribution over $\mathbf{u}_1$. (see section 1.8).

The LFADS generator with inferred input is thus described by the following procedure and equations. First an initial condition for the generator is sampled from the prior on $\mathbf{g}_0$

$$\hat{\mathbf{g}}_0 \sim P^{g_0}(\mathbf{g}_0) = \mathcal{N}(0, \kappa\mathbf{I}), \quad (1)$$

with κ a hyperparameter. At each time step $t = 1,\ldots,T$, an inferred input, $\hat{\mathbf{u}}_t$, is sampled from its prior and fed into the network, and the network is evolved forward in time,

$$\hat{\mathbf{u}}_t \sim \begin{cases} P^{u_1}(\mathbf{u}_1), & \text{if } t = 1 \\ P^u(\mathbf{u}_t|\mathbf{u}_{t-1}), & \text{otherwise} \end{cases} \quad (2)$$

$$\mathbf{g}_t = RNN^{gen}(\mathbf{g}_{t-1}, \hat{\mathbf{u}}_t) \quad (3)$$

$$\mathbf{f}_t = \mathbf{W}^{fac}(\mathbf{g}_t) \quad (4)$$

$$\mathbf{r}_t = \exp(\mathbf{W}^{rate}(\mathbf{f}_t)) \quad (5)$$

$$\hat{\mathbf{x}}_t \sim Poisson(\mathbf{x}_t | \mathbf{r}_t). \quad (6)$$

Here "Poisson" indicates that each component of the spike vector $\mathbf{x}_t$ is generated by an independent Poisson process at a rate given by the corresponding component of the rate vector $\mathbf{r}_t$. The prior for both $\mathbf{g}_0$ and $\mathbf{u}_1$ are diagonal Gaussian distributions. The prior for $\mathbf{u}_t$ with $t > 1$ is an auto-regressive Gaussian prior, with a learnable autocorrelation time and process variance (see section 1.8 for more details). We chose the Gated Recurrent Unit (GRU)[46] as our recurrent function for all the networks we use (see section 1.7 for equations), including RNN^{gen}. We have not included the observed data $\mathbf{a}$ in the generator model defined above, but this can be done simply by including $\mathbf{a}_t$ as an additional input to the recurrent network in equation 3. Note that doing so will make the generation process necessarily dependent on including an observed input. The generator model is illustrated in Supp Fig. 11. This diagram and the above equations implement the conditional distribution $P(\mathbf{x}|\mathbf{z}) = P(\mathbf{x}|\{\mathbf{g}_0, \mathbf{u}_{1:T}\})$ of the VAE decoder framework.

1.4 LFADS Encoder—The approximate posterior distribution for LFADS is the product of two conditional distributions, one for $\mathbf{g}_0$ and one for $\mathbf{u}_t$. Both of these distributions are Gaussian with means and diagonal covariance matrices determined by the outputs of the encoder or controller RNNs (see Supp Fig. 12 and below). We begin by describing the network that defines $Q^{g_0}\left(\mathbf{g}_0 \middle| \mathbf{x}, \mathbf{a}\right)$. Its mean and variance are given in terms of a vector $\mathbf{E}^{gen}$ by

$$\boldsymbol{\mu}^{g_0} = \mathbf{W}^{\mu^{g_0}}(\mathbf{E}^{gen}) \quad (7)$$

$$\boldsymbol{\sigma}^{g_0} = \exp\left(\frac{1}{2}\mathbf{W}^{\sigma^{g_0}}(\mathbf{E}^{gen})\right). \quad (8)$$

$\mathbf{E}^{gen}$ is obtained by running two recurrent networks over the data, bidirectionally. One RNN runs forward (from $t = 1$ to $t = T$) in time and the other RNN runs backwards (from $t = T$ to $t = 1$),

$$\mathbf{e}_t^{gen,b} = RNN^{gen,b}\left(\mathbf{e}_{t+1}^{gen,b}, \left[\mathbf{x}_t, \mathbf{a}_t\right]\right) \quad (9)$$

$$\mathbf{e}_t^{gen,f} = RNN^{gen,f}\left(\mathbf{e}_{t-1}^{gen,f}, \left[\mathbf{x}_t, \mathbf{a}_t\right]\right) \quad (10)$$

with $\mathbf{e}_{T+1}^{gen,b}$ and $\mathbf{e}_0^{gen,f}$ learnable biases. Once this is done, $\mathbf{E}^{gen}$ is the concatenation

$$\mathbf{E}^{gen} = \left[\mathbf{e}_1^{gen,b}, \mathbf{e}_T^{gen,f}\right]. \quad (11)$$

Running the encoding network both forward and backward in time allows $\mathbf{E}^{gen}$ to reflect the entire time history of the data $\mathbf{x}_{1:T}$ and $\mathbf{a}_{1:T}$. Finally, we sample initial conditions $\hat{\mathbf{g}}_0$ according to the following distribution

$$\hat{\mathbf{g}}_0 \sim Q^{g_0}\left(\mathbf{g}_0 \middle| \mathbf{x}, \mathbf{a}\right) = \mathcal{N}\left(\mathbf{g}_0 \middle| \boldsymbol{\mu}^{g_0}, \boldsymbol{\sigma}^{g_0}\right) \quad (12)$$

for a normal distribution with mean $\mu_i^{g_0}$ and standard deviation $\sigma_i^{g_0}$ for the i^{th} element of $\mathbf{g}_0$.

The approximate posterior distribution for $\mathbf{u}_t$ is defined in a more complex way that involves both a second set of forward-backward encoder RNNs and another RNN called the controller. The forward and backward encoder RNNs provide the input to the controller RNN, and are defined at time t with state variables $\mathbf{e}_t^{con,b}$ and $\mathbf{e}_t^{con,f}$ that are defined by equations identical to 9 and 10 (although with different trainable network parameters). Finally, the time-dependent input to the controller RNN is defined as

$$\mathbf{E}_t^{con} = \left[\mathbf{e}_t^{con,b}, \mathbf{e}_t^{con,f}\right]. \quad (13)$$

Rather than feeding directly into a Gaussian distribution, this variable is passed through the controller RNN, which runs forward in time with the generator RNN and also receives the latent dynamic factor, $\mathbf{f}_{t-1}$ as input,

$$\mathbf{c}_t = RNN^{con}\left(\mathbf{c}_{t-1}, \left[\mathbf{E}_t^{con}, \mathbf{f}_{t-1}\right]\right). \quad (14)$$

Thus, the controller is privy to the information about $\mathbf{x}_{1:T}$ and $\mathbf{a}_{1:T}$ encoded in the variable $\mathbf{E}_t^{con}$, and it receives information about what the generator network is producing through the latent dynamic factor $\mathbf{f}_{t-1}$. It is necessary for the controller to receive the factors so that it can correctly decide when to intervene in the generation process. Because $\mathbf{f}_{t-1}$ depends on both $\mathbf{g}_0$ and $\mathbf{u}_{1:t-1}$, these stochastic variables are included in the conditional dependence of the approximate posterior distribution $Q^u(\mathbf{u}_t|\mathbf{u}_{1:t-1},\mathbf{g}_0,\mathbf{x}_{1:T},\mathbf{a}_{1:T})$. The initial state of the controller network, $\mathbf{c}_0$, is defined as a trainable bias initialized to the 0 vector.

Finally, the inferred input, $\mathbf{u}_t$, at each time, is a stochastic variable drawn from a diagonal Gaussian distribution with mean and log-variance given by an affine transformation of the controller network state, $\mathbf{c}_t$,

$$\widehat{\mathbf{u}}_t \sim Q^u(\mathbf{u}_t|\mathbf{x},\mathbf{a}) = \mathcal{N}(\mathbf{u}_t|\boldsymbol{\mu}_t^u,\boldsymbol{\sigma}_t^u) \quad (15)$$

with

$$\boldsymbol{\mu}_t^u = \mathbf{W}^{\mu^u}(\mathbf{c}_t) \quad (16)$$

$$\boldsymbol{\sigma}_t^u = \exp\left(\frac{1}{2}\mathbf{W}^{\sigma^u}(\mathbf{c}_t)\right). \quad (17)$$

We control the information flow out of the controller and into the generator by applying a regularizer on $\mathbf{u}_t$ (a KL divergence term, described in Sections 1.6 and 1.10), and also by explicitly limiting the dimensionality of $\mathbf{u}_t$, the latter of which is controlled by a hyperparameter.

1.5 The full LFADS inference model—The full LFADS model (Supp Fig. 12) is run in the following way. First, a data trial is chosen, the initial condition and inferred input encoders are run, and an initial condition is sampled from the approximate posterior, $\widehat{\mathbf{g}}_0 \sim \mathcal{N}\left(\mathbf{g}_0\middle|\boldsymbol{\mu}^{g_0},\boldsymbol{\sigma}^{g_0}\right)$. Then, for each time step from 1 to T, the generator is updated, as well as the factors and rates, according to

$$\mathbf{c}_t = RNN^{con}(\mathbf{c}_{t-1},[\mathbf{E}_t^{con},\mathbf{f}_{t-1}]) \quad (18)$$

$$\boldsymbol{\mu}_t^u = \mathbf{W}^{\mu^u}(\mathbf{c}_t) \quad (19)$$

$$\boldsymbol{\sigma}_t^u = \exp\left(\frac{1}{2}\mathbf{W}^{\sigma^u}(\mathbf{c}_t)\right) \quad (20)$$

$$\hat{\mathbf{u}}_t \sim \mathcal{N}\left(\mathbf{u}_t \middle| \boldsymbol{\mu}_t^u, \boldsymbol{\sigma}_t^u\right) \quad (21)$$

$$\mathbf{g}_t = RNN^{gen}\left(\mathbf{g}_{t-1}, \hat{\mathbf{u}}_t\right) \quad (22)$$

$$\mathbf{f}_t = \mathbf{W}^{fac}(\mathbf{g}_t) \quad (23)$$

$$\mathbf{r}_t = \exp\left(\mathbf{W}^{rate}(\mathbf{f}_t)\right) \quad (24)$$

$$\hat{\mathbf{x}}_t \sim Poisson(\mathbf{x}_t | \mathbf{r}_t). \quad (25)$$

After training, the full model can be run, starting with any single trial or a set of trials corresponding to a particular experimental condition to determine the associated dynamic factors, firing rates and inferred inputs for that trial or condition. This is done by averaging over several runs to marginalize over the stochastic variables $\mathbf{g}_0$ and $\mathbf{u}_{1:T}$. Typically, equation 25 is not executed, unless one explicitly desires to generate spikes.

1.6 The loss function—To optimize our model, we would like to maximize the log likelihood of the data, $\sum_{\mathbf{x}} \log P(\mathbf{x}_{1:T})$, marginalizing over all latent variables. For reasons of intractability, the VAE framework is based on maximizing a variational lower bound, $\mathcal{L}$, on the marginal data log-likelihood,

$$\log P\left(\mathbf{x}_{1:T}\right) \geq \mathcal{L} = \mathcal{L}^x - \mathcal{L}^{KL}. \quad (26)$$

$\mathcal{L}^x$ is the log-likelihood of the reconstruction of the data, given the inferred firing rates, and $\mathcal{L}^{KL}$ is a non-negative penalty that restricts the approximate posterior distributions from deviating too far from the (uninformative) prior distribution. $\mathcal{L}^x$ and $\mathcal{L}^{KL}$ are then defined as

$$\mathscr{L}^{x} = \left\langle \sum_{t=1}^{T} \log\left(Poisson(\mathbf{x}_t|\mathbf{r}_t)\right) \right\rangle_{\mathbf{g}_0, \mathbf{u}_{1:T}} \quad (27)$$

$$\begin{aligned} \mathscr{L}^{KL} &= \left\langle D_{KL}\left(\mathcal{N}\left(\mathbf{g}_0 \middle| \boldsymbol{\mu}^{g_0}, \boldsymbol{\sigma}^{g_0}\right) \middle|\middle| P^{g_0}(\mathbf{g}_0)\right)\right\rangle_{\mathbf{g}_0} + \quad (28) \\ &= \left\langle D_{KL}\left(\mathcal{N}(\mathbf{u}_1|\boldsymbol{\mu}_1^u, \boldsymbol{\sigma}_1^u) \middle|\middle| P^{u_1}(\mathbf{u}_1)\right)\right\rangle_{\mathbf{g}_0, \mathbf{u}_1} + \\ &= \left\langle \sum_{t=2}^{T} D_{KL}\left(\mathcal{N}(\mathbf{u}_t|\boldsymbol{\mu}_t^u, \boldsymbol{\sigma}_t^u) \middle|\middle| P^u(\mathbf{u}_t|\mathbf{u}_{t-1})\right)\right\rangle_{\mathbf{g}_0, \mathbf{u}_{1:T}}, \end{aligned}$$

where the brackets denote marginalizations over the sub-scripted variables. Evaluating the $T + 1$ KL terms is done analytically for the Gaussian distributions and via sampling for the auto-regressive prior; the formulae for the Gaussians are found in Appendix B of (Kingma & Welling, 2013). We minimize the negative bound, $-\mathscr{L}$, using the reparameterization trick for Gaussian distributions to back-propagate low-variance, unbiased gradient estimates (Kingma & Welling, 2013). These gradients are used to train the system in an end-to-end fashion, as is typically done in deterministic settings.

1.7 GRU equations—For clarity, we use the common variable symbols associated with the GRU, with the understanding that the variables represented here by these symbols are not the same variables as those in the general LFADS model description. For $\mathbf{x}_t$ the input and $\mathbf{h}_t$ the hidden state at time t, the GRU update equation, $\mathbf{h}_t = GRU(\mathbf{x}_t, \mathbf{h}_{t-1})$, is defined as

$$\mathbf{r}_t = \sigma\left(\mathbf{W}^r\left(\left[\mathbf{x}_t, \mathbf{h}_{t-1}\right]\right)\right) \quad (29)$$

$$\mathbf{u}_t = \sigma\left(\mathbf{W}^u\left(\left[\mathbf{x}_t, \mathbf{h}_{t-1}\right]\right)\right) \quad (30)$$

$$\mathbf{c}_t = \tanh\left(\mathbf{W}^c\left(\left[\mathbf{x}_t, \mathbf{r}_t \odot \mathbf{h}_{t-1}\right]\right)\right) \quad (31)$$

$$\mathbf{h}_t = \mathbf{u}_t \odot \mathbf{h}_{t-1} + \left(1 - \mathbf{u}_t\right) \odot \mathbf{c}_t, \quad (32)$$

with $\odot$ denoting element-wise multiplication and σ denoting the logistic function.

1.8 Autogressive prior for inferred input—A zero-mean auto-regressive process with one time lag (AR(1)) is defined by

$$s(t) \quad = \alpha s(t-1) + \epsilon_s(t), \quad (33)$$

with $0 \leq \alpha < 1$ and noise variable $\epsilon_s(t)$ drawn from $\mathcal{N}\left(0, \sigma_\epsilon^2\right)$. An equivalent formulation for AR(1) process is to define α and σ_ϵ^2 in terms of a process autocorrelation, τ, and process variance, σ_p^2, as $\alpha = \exp(-1/\tau)$ and $\sigma_\epsilon^2 = \sigma_p^2\left(1-\alpha^2\right)$. To make the process distribution stationary the correct distribution for $s(0)$ is $\mathcal{N}\left(0, \sigma_p^2\right)$. Applying this to LFADS, the prior for $\mathbf{u}_t$ with $t > 1$ is an independent AR(1) process in each dimension, such that for the i^{th} element of $\mathbf{u}_t$ an autocorrelation τ_i and process variance $\sigma_{p,i}^2$ are initialized to user-defined initial values.

1.9 Modifications to the LFADS algorithm for stitching together data from multiple recording sessions—To accommodate multiple recordings sessions, as in Fig. 5 of the main text, we make minor modifications to the LFADS architecture. In particular, we allow each separate recording session to have unique "read-in" and "read-out" adaptor matrices. The reasons are both practical and conceptual. Practically, a different number of units are recorded in each recording session; consequently, the number of inputs and outputs to the LFADS algorithm needs to change accordingly. Conceptually, the hypothesis of most investigators when recording in the same area across multiple sessions is that they are recording different measurements of the same underlying (dynamical) system. Therefore, LFADS allows a different input and output transformation for each recording session to handle the different measurements, but otherwise LFADS models all the data with the same generative model, with shared parameters across all recording sessions, to allow different sessions' measurements to improve the underlying model. The encoder network, generator network, and matrix $\mathbf{W}^{fac}$ mapping from generator units to factors remain shared.

Beginning with the simpler case of the read-out matrices, which map from factors to recorded units, we modify equation 5, replacing it with equation 12 to change matrices as a function of recording session, thus introducing a session index, s, into the notation

$$\mathbf{r}_{s,t} \quad = \exp\left(\mathbf{W}_s^{rate}\left(\mathbf{f}_{s,t}\right)\right), \quad (34)$$

where the dimensions of $\mathbf{W}_s^{rate}$ are now the number of units in the session, D_s, by the number of factors in the LFADS model, F, the latter of which is independent of the session.

We now address the read-in matrices. Without multiple recording sessions, we simply feed the recorded spikes, $\mathbf{x}_t$ into the encoders (equations 9 and 10), single trial by single trial. To handle multiple sessions' data, we modify this practice by introducing a per-session read-in matrix, $\mathbf{W}_s^{input}$. These read-in matrices map from recorded units to "input factors". Then, for

the bidirectional encoding RNN for $\mathbf{g}0$, we modified equations 9 and 10 by inputting the linearly transformed spikes, yielding

$$\mathbf{e}_{s,t}^{gen,b} = RNN^{gen,b}\left(\mathbf{e}_{s,t+1}^{gen,b}, \left[\mathbf{W}_s^{input}\left(\mathbf{x}_{s,t}\right), \mathbf{a}_t\right]\right) \quad (35)$$

$$\mathbf{e}_{s,t}^{gen,f} = RNN^{gen,f}\left(\mathbf{e}_{s,t-1}^{gen,f}, \left[\mathbf{W}_s^{input}\left(\mathbf{x}_{s,t}\right), \mathbf{a}_t\right]\right), \quad (36)$$

where the dimensions of $\mathbf{W}_s^{input}\left(.\right)$ are $F \times D_s$. We modify the bidirectional RNN encoder for input to the RNN controller in the same way. Otherwise the LFADS architecture is identical to the standard use case, with the rest of the parameters of the LFADS architecture shared across all recording sessions.

We computed appropriate initial parameter settings for both the read-in and read-out matrices using a principal components regression technique. Briefly, we assembled a matrix of within-condition averaged firing rates for each unit across all sessions, with dimension equal to the total number of units x number of time points, $\sum_s D_s \times T$. We performed principal components analysis on this matrix to reduce it to F principal components, equivalent to the number of factors in the model, yielding a F x T matrix of principal component (PC) scores. For each session, we regressed the matrix of PC scores against the condition-averaged firing rates recorded in that session. The resulting matrix of regression coefficients, which best reconstructs a set of shared PC scores from each session's firing rates, was used as the initial read-in matrix for that session, $\mathbf{W}_s^{input}$. The read-out matrix $\mathbf{W}_s^{rate}$ for the session was initialized to the pseudoinverse of $\mathbf{W}_s^{input}$. These read-in and read-out matrices can be thought of as seeding the multi-session LFADS model with a correspondence across recording sessions. The read-in and read-out matrices are learned as parameters from the data from the corresponding session simultaneously with the shared parameters. Optionally, the read-in matrices can be treated as fixed to encourage that LFADS to use a consistent representation of similar trials, e.g. trials from the same behavioral condition. We used this fixed read-in matrix approach in the dynamical stitching example in the main text.

We train the model by selecting one dataset at a time at random (e.g. the first session), and the correct read-in and read-out matrices are then used (the matrices associated with the first session). To generate a mini-batch of gradients, the algorithm then selects a random mini-batch of data from that session and propagates it forward to evaluate the loss. The relevant gradients of the loss are then back-propagated. As a result, all shared parameters (e.g. the encoder and generator RNN parameters and factor read-out matrix $\mathbf{W}^{fac}$) are modified with every mini-batch of data regardless of dataset, while the read-in and read-out matrices are modified only when data from that session is used for training.

1.10 Hyper-parameters and further details of LFADS implementation.—A table of the major hyper-parameters for each model is listed in Methods Table 1. There were a number of additional standard details that aided in the optimization and generalization of the LFADS model applied to the datasets in our study.

- For all models, the time step of the LFADS RNNs was equal to the data bin size.
- To help avoid over-fitting, we added a dropout layer[47] to the inputs and to a few feed-forward (input) connections[48] in the LFADS model. Specifically, we used dropout "layers" around equation 11, around the input in equation 18, and around equation 22.
- We added an L_2 penalty to recurrent portions of the generator (equations 29–32) and controller networks to encourage simple dynamics. Specifically, we regularized any matrix parameter by which $\mathbf{h}_{t-1}$ was multiplied, but not those that multiplied $\mathbf{x}_t$.
- As defined in eqn. 28, there is an information limiting regularizer placed on $\mathbf{u}_t$ by virtue of minimizing the KL divergence between the approximate posterior over $\mathbf{u}_t$ and the uninformative auto-regressive prior.
- Following [49], we added a linearly increasing schedule on the KL divergence penalty so that the optimization does not quickly (and pathologically) set the KL divergence to 0. By 2000 training steps, the schedule reached the maximum value of the KL penalty. An identical schedule was used for linearly increasing the L_2 regularizer on the network parameters.
- We experimented with the variance of the prior distribution for the initial condition distribution and settled on a value of $\kappa = 0.1$, chosen to avoid saturating network nonlinearities.
- The auto-regressive prior parameters were optimized to reduce the KL divergence between inferred inputs from the approximate posterior distributions and those of the prior. In practice, nearly all AR(1) processes optimized to the uncorrelated, white noise case ($\tau_i \approx 0$ and $\sigma^2_{p,i} \approx \sigma^2_{\epsilon,i} \approx 0.1$). We initialized them with $\tau_i = 10$ time steps and $\sigma^2_{\epsilon,i} = 0.1$.
- Unless otherwise specified, all matrices were randomly initialized with a normal distribution with mean equal to 0, and variance equal to $1/K$, where K is input dimension of the matrix. All biases were initialized to 0.
- We used the ADAM optimizer, with initial learning rate of 0.01, and $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 0.1$. During training, the learning rate was decreased whenever the training error for the current epoch of data was greater than the last 6 training error values. In this case, the learning rate was decayed by multiplying the rate by 0.95, and 6 training epochs were required before the learning rate could be decayed again. The optimization continued until the learning rate was less than or equal to 1e-5. We routinely saved checkpoints of the model and therefore were able to capture the model with the lowest validation error.

- We clipped our hidden state $\mathbf{h}_t$ when any of its values went above a set threshold. This threshold was rarely hit, but was useful to avoid occasional pathological conditions.
- We used gradient clipping with a value of 200 to avoid occasional pathological gradients.
- The matrix in the $\mathbf{W}^{fac}(\cdot)$ affine transformation was row-normalized to keep the factors relatively evenly scaled with respect to each other.
- To monitor overfitting, a portion of the data is set aside as a validation set, and these data are never used to update the model's weights. Instead they are simply used to evaluate reconstruction cost on held-out data. For all analyses in this manuscript, we used a ratio of 4:1 between training and validation data.

1.11 Computing posterior averages of model variables.—As the LFADS model is inherently stochastic, one needs to average over draws of the latent variables to get good estimates of meaningful quantities within the network (e.g. the rates, $\mathbf{r}_t$). For example, in denoising a single trial of spike trains, we run the full LFADS model - both encoder and decoder on the single trial. For that single trial, we sample the stochastic variables, (eqns. 12 and 15) some number of times (e.g. 512) and then evaluate the generative portion of the model with these sampled variables. Finally, we obtain the mean of the quantity, in this case, the posterior average, computed by averaging the quantity of interest over the random samples of the stochastic variables, e.g. $\bar{\mathbf{r}}_t \equiv \langle \mathbf{r}_t \rangle_{\mathbf{g}_0, \mathbf{u}_{1:T}}$. It is posterior averages such as $\bar{\mathbf{r}}_t$ that are shown in the majority of figures.

1.12 LFADS related work in machine learning literature—Recurrent neural networks have been used extensively to model neuroscientific data (e.g. [2,7,50–52]), 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 [23,43], 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[53], deep Kalman filters[45], and the RNN DRAW network[44].

There is also a line of research applying probabilistic sequential graphical models to neural data. Recent examples include PLDS[17], switching LDS[19], GCLDS[54], and PfLDS[15]. 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[12] 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 [13] 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 [55]. The authors of [45] have defined a very general nonlinear variational sequential model, which they call the Deep Kalman Filter (DKF). The authors of [56] applied recurrent variational architectures to problems of control from raw images. Finally, [57] 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.

2 Synthetic datasets

2.1 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., [13,18]), 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.

2.2 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

$$\dot{y}_1 = \sigma(y_2 - y_1) \quad (37)$$

$$\dot{y}_2 = y_1(\rho - y_3) - y_2 \quad (38)$$

$$\dot{y}_3 = y_1 y_2 - \beta y_3 . \quad (39)$$

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 [13], 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 [13], we purposefully chose parameters that made the dataset more challenging. Specifically, relative to [13], 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.

2.3 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,

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

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. [2,7,50–52]). 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 [50] 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).

2.4 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 Methods 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 $\mathbf{B}$ 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 $\bar{\mathbf{u}}_t = \langle \mathbf{u}_t \rangle_{\mathbf{g}_0, \mathbf{u}_{1:T}}$. 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 $\bar{\mathbf{u}}_t$ reached its maximum value.

2.5 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 ($\bar{\mathbf{u}}_t$) for a given trial were extracted by taking 1024 samples from the ($\bar{\mathbf{u}}_t$) 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.

3 Neural datasets - Research participants with paralysis

Permission for these studies was granted by the US Food and Drug Administration (Investigational Device Exemption) and Institutional Review Boards of Stanford University (protocol # 20804), Partners Healthcare/Massachusetts General Hospital (2011P001036), Providence VA Medical Center (2011–009), and Brown University (0809992560). The participants in this study were enrolled in a pilot clinical trial of the BrainGate Neural Interface System (http://www.clinicaltrials.gov/ct2/show/NCT00912041). Informed consent, including consent to publish, was obtained from the participants prior to their enrollment in the study.

Participant T7 was a right-handed man, 54 years old at the time of the research sessions reported here, who was diagnosed with ALS and had resultant motor impairment (ALSFRS-R of 17). In July 2013, participant T7 had two 96-channel intracortical silicon micro-electrode arrays (1.5 mm electrode length, Blackrock Microsystems, Salt Lake City, UT) implanted in the hand area of dominant motor cortex. T7 retained very limited and inconsistent finger movements. Data reported are from T7's post-implant day 231.

A second study participant, T5, is a right-handed man, 63 years old at the time of the research sessions reported here, with a C4 ASIA C spinal cord injury that occurred approximately 9 years prior to study enrollment. He retains the ability to weakly flex his left (non-dominant) elbow and fingers; these are his only reproducible movements of his extremities. He also retains some slight residual movement which is inconsistently present in both the upper and lower extremities, mainly seen at ankle dorsiflexion and plantarflexion, wrist, fingers and elbow, more consistently present on the left than on the right. Occasionally, the initial slight voluntary movement triggers involuntary spastic flexion of the limb. In Aug. 2016, participant T5 had two 96-channel intracortical silicon micro-electrode

arrays (1.5 mm electrode length, Blackrock Microsystems, Salt Lake City, UT) implanted in the upper extremity area of dominant motor cortex. Data reported are from T5's post-implant day 51.

3.1 Task design and data analysis—Neural data were recorded during "Center-out-and-back" target acquisition tasks. The data were originally collected for neural prosthetic decoder calibration, as part of research testing algorithms for closed-loop neural cursor control[8,34,35]. In the Center-out-and-back task, data were collected either in motor-based control (with T7, who retained limited residual movements), or an attempted movement paradigm (with T5, who did not retain sufficient movement to reliably measure or physically control a cursor). In motor-based control, T7 controlled the position of a cursor on a computer screen by making physical movements with his fingers on a wireless touch-pad (Magic Trackpad; Apple, Cupertino, CA). The cursor began in the center of the screen, and targets would appear in one of 8 locations on the periphery. The participant then acquired the targets by moving the cursor over the target and holding it over the target for 500 ms. Participant T7's limited movements spanned a small region on the touch-pad, approximately 1/8"–1/4" wide. In the attempted movement paradigm, the cursor was automatically moved directly toward the target by the computer, and T5 was asked to attempt movements of his whole arm that followed the movements of the cursor.

Voltage signals from each of the electrodes were band-pass filtered from 250 to 7500 Hz and then processed to obtain multi-unit 'threshold crossings,' i.e., discrete events that occurred whenever the voltage crossed below a threshold (choice of threshold was dependent on the array- T7 lateral array: -80 μV; T7 medial array: -95 μV; T5, both arrays: -3.5 times the r.m.s. voltage on each channel.). For the present analyses, we did not "spike sort" and instead grouped together threshold crossings on a given electrode. These spikes therefore can include both single- and multi-unit activity. For both participants, analysis was restricted to channels known to show significant modulation during movement attempts (T7: 78 channels; T5: 187 channels).

Neural control and task cueing were controlled by custom software run on the Simulink/xPC real-time platform (The Mathworks, Natick, MA), enabling millisecond-timing precision for all computations. Neural data collected by the NeuroPort System (Blackrock Microsystems, Salt Lake City) were available to the real-time system with 5-ms latency. Visual presentation was provided by a computer via a custom low-latency network software interface to Psychophysics Toolbox for MatLab and an LCD monitor with a refresh rate of 120 Hz. Frame updates from the real-time system occurred on screen with a latency of approximately 7 ± 5 ms.

4 Neural datasets - Nonhuman primates

All procedures and experiments were approved by the Stanford University Institutional Animal Care and Use Committee.

4.1 Maze task—An adult male macaque monkey (monkey J) was trained to sit head-fixed in a primate chair and perform 2D target acquisition tasks in a fronto-parallel plane by controlling an on-screen cursor with his hand movements. Monkey J was implanted with two

96-electrode arrays (1 mm electrodes spaced 400 μm apart, Blackrock Microsystems) using standard neurosurgical techniques. The arrays were implanted into M1 and dorsal premotor cortex (PMd) of the hemisphere contralateral to his reaching arm.

The Maze task is a variant of a center-out delayed reach task, whose details have previously been described [21]. Briefly, monkey J made arm movements in a 2-dimensional workspace while the position of the right index and middle fingertips was tracked optically. This tracked position controlled the movements of a virtual cursor, and the cursor's position floated 2.5 cm above the hand. To initiate a trial, the monkey fixated on a fixation spot for >400 ms, after which a target appeared. After a delay period (varying from 0 – 900 ms), a go cue instructed the monkey to begin his movement. A set of virtual barriers in the workspace facilitated the instruction of curved or straight reach trajectories. Contact with a barrier resulted in an unrewarded trial. A trial was counted as a success, and reward delivered, if the monkey held the cursor on the target for 450 ms.

Several de-noising methods were applied to the Maze dataset. For all methods, individual trials were aligned to movement onset (the point at which movement is first detectable), and data consisted of 450 ms preceding and following movement onset (for a total of 900 ms per trial). The dataset consisted of 2296 trials across 108 different reach conditions (target and barrier locations), and 202 single units were isolated from the recorded activity.

For the temporal and neural cross correlation matrices (Supp. Data 2,3), neural activity was first condition-averaged such that the data formed a tensor, $X \in \mathbb{R}^{T \times N \times C}$, spanning T time points, N neurons, and C conditions. As before, trials were aligned to movement onset prior to averaging, and data consisted of the 450 ms preceding and following movement onset (for a total of 900 ms). For a given condition c, temporal cross correlation matrices were calculated as follows:

$$\Sigma_T^c = \sum_{n=1}^{N} X\left(:, n, c\right) X\left(:, n, c\right)^T \quad (41)$$

Similarly, for a given condition c, neural cross correlation matrices were calculated as follows:

$$\Sigma_N^c = \sum_{t=1}^{T} X\left(t, :, c\right) X\left(t, :, c\right)^T \quad (42)$$

Neural cross correlation matrices were sorted using a MATLAB implementation of the Bron-Kerbosch maximal clique algorithm (https://www.mathworks.com/matlabcentral/fileexchange/30413-bron-kerbosch-maximal-clique-finding-algorithm). To apply the algorithm, the cross correlation matrix for each condition was first converted into a sparse binary matrix by applying a 95% threshold (all values about the 95% percentile were set to 1, and all values below were set to 0). Applying the Bron-Kerbosch algorithm resulted in a

grouping of neurons by similarity, which was then used to sort the cross correlation matrix for each condition.

4.2 Center-out and Cursor Jump tasks—These experiments were also performed with Monkey J. Experiments were controlled using custom MATLAB and Simulink Realtime software (Mathworks, USA). Arm reaches were made with the display blocking the monkey's view of his hand. The task was displayed in virtual reality using a Wheatstone stereograph with a latency of 7 ± 4 ms as described in [58]. The virtual computer cursor followed the velocity of a reflective bead taped to the monkey's hand, which was tracked via an infrared system at 60 Hz (Polaris, Northern Digital, Canada). The non-reaching arm was gently restrained. To successfully acquire a target, the monkey had to hold the cursor within a 4 × 4 cm target acquisition area for a continuous 500 ms. A target color change cued that the cursor was within the acquisition area. If the cursor left the target area during this hold period, the 500 ms timer reset. The monkey had to acquire the target within a time limit of 2 seconds to receive a liquid reward and success tone.

Voltage signals from each of the electrodes were band-pass filtered from 250 to 7500 Hz and then processed to obtain multi-unit 'threshold crossings', i.e. discrete events that occurred whenever the voltage crossed below a threshold (set at the beginning of each day to be -4.5 times r.m.s. voltage). For the "Center-out-and-back" and "Cursor Jump" tasks, we did not spike sort the data and instead grouped together threshold crossings on a given electrode. These threshold crossing events therefore can include both single- and multi-unit activity.

For the LFP (Fig. 4 of main text) and Cursor Jump analyses (Fig. 6 of main text), data analyzed were from dataset 2015–04-15, which occurred 69 months after the implantation of recording arrays. A single LFADS model was fit to data from two types of reaching tasks - a standard "Center-out-and-back" task and a Cursor Jump task.

In the Center-out-and-back task, targets alternated between being located at the workspace center or at a randomly chosen target out of 8 possible target locations, all 12 cm away from the workspace center and evenly spaced around a circle. In the Cursor Jump task, targets were located either at the workspace center or one of two radial target locations located 12 cm away from the workspace center, in opposite directions. The three possible targets lay along the vertical monitor axis.

The 'cursor jump' manipulation at the heart of the Cursor Jump Task was applied on a random 25% of trials towards radial targets. On these randomly selected perturbation trials, during the monkey's reaching movement, the cursor position jumped, i.e., it was offset by 6 cm perpendicular to the vertical axis. The jump happened after the cursor traveled 6 cm towards the target along the vertical axis. Only one perturbation occurred per trial. The time when the cursor jump command was sent to the display computer was recorded with 1 ms resolution, after which it appeared at the next 120 Hz monitor update. The delivery of cursor jump position offsets required us to counteract this offset at the end of each perturbed outward trial so as to not carry a (possibly accumulating) hand-to-cursor offset over multiple trials. Thus, we applied a second, opposite cursor jump as soon as the center target re-

appeared, resulting in a consistent hand-to-cursor position relationship at the start of each outward trial.

To train the LFADS model, spike trains were binned at 10 ms resolution. A single LFADS model was fit to a combined dataset containing center-out-and-back trials (8 targets), outward trials without perturbations (2 targets), outward trials with perturbations (2 targets, 2 perturbation directions), and return-to-center trials from the perturbed/unperturbed outward trials, for a total of 5140 trials. 800 ms of data were taken for each trial, with data aligned to the start of the trial (target onset). In cases of perturbations, most jumps happened between 400–550 ms post-target onset. The model was allowed to infer 4 inputs to the generator in order to fit the data. The choice of 4 inputs reflects three key facts about the system and task. First, we know that high-frequency oscillatory dynamics are present in the firing rates (Fig. 6), which require inputs to model. In this particular dataset, we recorded from electrode arrays in two different brain areas (M1/PMd), which exhibit different oscillations, and thus we needed two inputs to model these features. Second, there are specific task-related perturbations that we must model: prior to target onset, the subject does not know whether an upward or downward target will appear. Thus the arrival of target position information to motor cortex is a 1-dimensional perturbation (upwards or downwards) that occurs early in the trial. Third, during the actual reaching movement, a left or right perturbation may occur with low probability. This provided a separate 1-dimensional perturbation for the system to model. Thus, we reasoned that 4 inputs was a reasonable choice for modeling this particular recording configuration and task.

4.3 Multi-session V-probe recordings—One adult male macaque monkey (P) was trained in a behavioral task as described below. After initial training, we performed a sterile surgery during which the macaque was implanted with a head restraint and a recording cylinder (NAN Instruments), which was located over left, caudal, dorsal premotor cortex (PMd). The cylinder was placed surface normal to the skull and secured with methyl methacrylate. A thin layer of methyl was also deposited atop the intact, exposed skull within the chamber. Before recording sessions began, a miniature craniotomy (3 mm diameter) was made under ketamine/xylazine anesthesia, targeting an area in PMd which responded during movements and palpation of the upper arm (17 mm anterior to interaural stereotaxic zero).

In the behavioral task, monkey P was trained to use his right hand to grasp and translate a custom 3D printed handle (Shapeways, Inc.) attached to a haptic feedback device (Delta.3, Force Dimension, Inc.). The other arm was comfortably restrained at the monkey's side. The haptic device was controlled via a 4-poll position, update force feedback loop implemented in custom software written in C++ atop Chai3D (http://chai3d.org). The weight of the device was compensated by upward force precisely applied by the device's motors, such that the motion of the device felt nearly effortless because the device's mechanical components were lightweight and had low inertia. The device endpoint with the attached monkey handle was constrained via software control to translate freely in the fronto-parallel plane. The handle was custom 3D printed and contained a beam break detector which indicated whether the monkey was gripping the handle. The task was controlled using custom code running on a dedicated computer running the Simulink Real-Time operating system. Hand position was recorded at 1 kHz, and the 2D position of the device was used to update the position of a

white circular cursor at the refresh rate of 144 Hz with a latency of 4–12 ms (verified via photodiode) displayed on an LCD screen located in front of the monkey and above the haptic device, in the same fronto-parallel plane as the device itself. The display was driven by custom software driven by Psychophysics Toolbox. A plastic visor was used to mask the monkey's visual field such that he could see the screen but not his hand or the haptic device handle.

The monkey was trained to perform a delayed center-out reaching task by moving the haptic device cursor towards visual targets displayed on the screen. Monkeys initiated the task by holding onto the device handle, which was detected by a beam break photodiode built into the handle. At the start of each trial, the device gently returned the hand to the center position and supported the weight of the arm from below in that position (by rendering a narrow virtual shelf just below the haptic cursor). At *target onset*, one or more reach targets appeared as hollow circles at one of 8 radial locations located 10 cm from the position. After a variable delay period (50–800 ms), the *go cue* was indicated visually by the target outline filling in with color. A trial was successful if the monkey remained still during the delay period, initiated the reach within 600 ms after the go cue, and held in the reach target for 50 ms. In some sessions, the monkey performed additional trial conditions with different target locations or forces applied to the haptic device. These trials were excluded from analysis; only successful center-out reaches were included. Hand velocities were computed by applying a smoothing, differentiating filter (Savitzy-Golay, 2nd-order, 3 ms smoothing widow) to the raw position time series. Reaction time was measured from the visual display of the go cue detected at the photodiode until the hand speed in the fronto-parallel plane reached 5% of the peak speed on each trial.

Electrophysiological recordings were performed by slowly lowering a linear multielectrode array with 24 recording channels (Plexon V-probe or U-probe) to a position where the channels likely spanned the layers of the cortex based on properties of the neural signals. We allowed 45–90 minutes to allow the probe to settle before beginning experiments. All 24 channels were amplified and sampled at 30 kHz (Blackrock Microsystems), high-pass filtered (fourth-order Butterworth filter, 250 Hz corner frequency), and thresholded at −3.5× RMS voltage on each channel. Threshold crossings on adjacent channels that occurred within 0.5 ms of each other were removed from one of the channels to avoid duplicate detection of spiking along the array. Threshold crossing rates were then binned at 10 ms on each channel.

Experimental sessions were screened based on minimum trial count (200 trials); one dataset was manually excluded based on an abrupt discontinuity in the recorded firing rates over the session. Following this screening, a total of 44 consecutive experimental sessions were included, comprising recording locations in the upper arm representation of primary motor cortex and dorsal premotor cortex. A 1200 ms time window beginning 500 ms before the go cue to 700 ms afterwards was chosen from each successful trial and used to train the LFADS model.

5 Analysis Methods by Figure

We used a number of analysis methods on either smoothed neural data, or the output of LFADS, typically the rates, factors or inferred inputs. All of these analyses methods are standard, but we provide references and operating parameters here.

5.1 Figure 2 - Application of LFADS to a "Maze" reaching task

For the PSTHs and single-trial inferred/estimated firing rates in Fig. 2b, each trial was aligned by movement onset (i.e., the time at which movement of the arm is first detectable), and data analyzed began 400 ms prior to movement onset and ended 400 ms after movement onset. Data were pre-processed via one of three techniques: Gaussian smoothing, GPFA[12], or LFADS. For Gaussian smoothing, the millisecond-binned spike trains were convolved with a Gaussian function with standard deviation (s.d.) of 30 ms - this parameter was optimized to produce PSTHs with visible structure, while preserving some of the fast-timescale features seen in the neural firing rates. For GPFA, the number of latent factors was 40, and the binsize was 20 ms, optimized as mentioned below. For LFADS, the binsize was 5 ms, and the number of latent factors was fixed at 40.

For the t-SNE analysis (**Fig. 2c,** Supp. Video 1) the initial conditions vector inferred by LFADS (g0) for each trial was mapped onto a low-dimensional subspace using t-SNE[59]. 3 dimensions were used, and the perplexity parameter was set to 75 (similar results were obtained with a wide variety of parameters). Points were plotted in the 3-D t-SNE space, with colors corresponding to the endpoint of the reaching movement (Fig. 2a), and marker type corresponding to the type of reach (i.e., markers used were circles, squares, and triangles, for straight, curved counter-clockwise, and curved clockwise reaches, respectively).

To decode kinematics from neural features (Figs. 2d,e), we used Optimal Linear Estimation[26] to create decoders that mapped neural features onto the measured x and y reaching velocities. The inputs to the decoder were the raw or de-noised neural data from 250 ms prior to 450 ms post movement onset. De-noising was achieved via one of three techniques mentioned previously. For Gaussian smoothing, a 40 ms s.d. was used (optimized via cross-validated decoding). For GPFA, the number of latent factors and binsize was optimized to maximize decoding accuracy, with binsize swept from 5–20 ms, and latents swept from 5–40 factors (see Supp. Fig. 4 for results of the optimization on the full population of neurons). For LFADS, the binsize was fixed at 5 ms, and the number of latents was set to 40 for the full population, and 20 for the subsampling analysis (next paragraph). The neural features from each technique were the Gaussian-smoothed firing rates, factor estimates using GPFA, or de-noised firing rates using LFADS. In all cases, to decode kinematics, the neural features were 'lagged' by 90 ms to account for delays between neural activity and measured kinematics (optimized using cross-validated decoding), and the neural features were binned at 20 ms as a standard for comparison.

Kinematic predictions were generated using 5-fold cross-validation. The subsampling analyses followed the above, with limited populations achieved via random subsampling (without replacement) from the full population of 202 neurons. Decoding performance was

quantified using goodness of fit (R^2) between the original and reconstructed velocities (validation trials from the 5-fold cross-validated decoding) for the x and y dimensions. For the sample reconstructed reach trajectories shown in Fig. 2d, trajectories were seeded with the true initial position, and subsequent points in the trajectory were calculated by simply integrating the decoded velocity at each timestep.

Note that for offline decoding analyses, the approach of smoothing neural data and then linearly regressing against kinematics, outlined here, is a generalization of common brain-machine interface (BMI) decoding approaches such as the Kalman Filter. This relationship is outlined in [27]; briefly, the Kalman filter can be viewed as a two-step process - first smoothing the data, and subsequently performing a linear dimensionality reduction step that maps the smoothed, high-dimensional neural data onto kinematics. In the Kalman Filter the amount of smoothing is largely determined by the simple linear dynamical system (LDS) that models state evolution (i.e., models changes in kinematics). This can be especially problematic in datasets with highly varied kinematics, such as the complex "maze" reaching dataset, where a simple LDS does not provide a good model of observed kinematics. Therefore, to avoid having the degree of smoothing influenced by a poorly-fit kinematics model, we optimized the smoothing parameter using cross-validated decoding as described above.

Further improvement can be achieved for online (closed-loop) BMI control using an additional "intention estimation" step, and then regressing neural data against the inferred intention rather than the measured kinematics. This "intention estimation" step has been shown to improve closed-loop BMI control when intention is estimated from hand reaching data (e.g., the FIT Kalman Filter[60]) or estimated from closed-loop BMI control (e.g. the Re-FIT Kalman Filter[34,58]). However, to date, these approaches having been applied to simple datasets (point-to-point movements) to calibrate BMI decoders, and assume that the subject's intention was to move in a straight line toward the target. In the complex "maze" dataset analyzed in Fig. 2, the monkey made curved reaches which violate this assumption - therefore our decoding approach used regression against measured kinematics rather than attempting to infer the subject's intention.

For the held-out neuron analysis (Fig. 2f), we compared the accuracy of LFADS against GPFA in predicting held-out neurons in the Maze dataset. As in Fig. 2e, we sub-sampled neurons from the complete neural population (202 neurons total), and used the sub-sampled populations to estimate latent dynamics (25, 100, or 150 neurons to fit either LFADS or GPFA latent models; the same populations of neurons for the previous decoding analysis were used). We used a standard Generalized Linear Model (GLM) framework (Paninski, 2004) to map the latent state estimates produced by LFADS or GPFA onto the binned spike counts (20 ms bins) for the remaining held-out neurons, e.g., for a model trained with 25 neurons, there are 177=202–25 held out neurons. We then measured the improvement produced by the LFADS latent estimates over GPFA (evaluated using log likelihood per spike, (LLPS[42])). For a given held-out neuron, we predicted the neuron's firing rate based on the GLM fit, for all trials that were held out from the GLM fit. We then evaluated the LLPS of the observed spike trains given the predicted firing rates. For almost all held-out neurons,

LFADS-inferred latent state estimates were much more predictive about the spike counts of the held-out neurons than estimates produced by GPFA.

5.2 Figure 3 - Rotations in state space

Rotations in state space were found using the jPCA technique, whose mathematical details are presented elsewhere[3]. We briefly summarize the overall approach here. jPCA was applied in two ways: first to examine rotations in the condition-averaged responses, and subsequently for the single-trial responses. For condition-averaged responses, each neuron's response was first averaged across all trials of the identical condition to create a set of condition-averaged firing rates. These firing rates were smoothed via convolution with a Gaussian kernel, with the width of the kernel chosen to reduce the noise in the firing rates without smoothing away the rotational content. Smoothed firing rates were then mean-centered across conditions at every time point by subtracting the average across-condition response from the response of each individual condition. The mean-centered rates were then "soft-normalized"[3] to prevent individual neurons (e.g. high firing rate or potentially noisy neurons) from dominating the results of the subsequent dimensionality-reduction step. These high-dimensional neural firing rates were projected into a low-dimensional subspace using PCA. Within this subspace (neural state space), we then used the jPCA technique to find planes that are best fit by a linear dynamical system with purely rotatory dynamics.

For the subsequent single-trial responses, the goal was to examine the same rotations in state space that were found via condition averaging, but examine their consistency at the level of single trials. Therefore, the single-trial data was projected into jPCA planes via the projections that were calculated in the condition-averaged analysis.

For monkey J, all trials were aligned to movement onset. We used 250 ms for jPCA analysis, with the time window starting 60 ms prior to movement onset. Observed neural firing rates were smoothed with a 40 ms s.d. Gaussian kernel to reduce noise, and soft-normalized with a value of 0.1. For the de-noised LFADS data, further smoothing and de-noising had little effect, so the parameters used were a 25 ms s.d. Gaussian kernel with a negligible soft-normalization value (5e-5). For the initial dimensionality-reduction step (PCA), 10 PCs were kept and used for jPCA.

As with the monkeys, the rotations in state space for research participants with paralysis are found by identifying the time period starting just before the rapid change in neural activity that occurs with a movement attempt[8]. For participant T5, because no movement was measurable, data were simply aligned to the start of the trial (i.e., the point at which targets are displayed). The window taken for jPCA analysis was 400 ms of data beginning 240 ms after the start of the trial. As with the monkey data, larger parameters for smoothing and greater soft-normalization were used to de-noise the observed neural responses, vs. the LFADS de-noised neural responses. These were a Gaussian kernel s.d. and soft-normalization parameter of 40 ms and 10 for the observed responses, and 25 ms and 5 for the LFADS de-noised neural responses.

For the held-out conditions analysis (Fig. 3i-k) - conditions were binned by dividing their endpoint (reach target) into 32 evenly-spaced angular bins. Because some angular bins did

not contain any targets, this resulted in 19 sets of reaching conditions. For each reaching condition set, a separate LFADS model was trained. In the initial training run, trials from all conditions not in the given angular bin were used to train the full LFADS model. After this initial training run, all model parameters beginning at the initial conditions vector were fixed (i.e., all weights that map from the IC vector to the generator, all internal weights of the generator, all read-out weights to the factors layer, all read-out weights to the individual neurons, and all bias terms). Fixing these parameters essentially locks the dynamics of the LFADS model (i.e., the dynamics of the generator) to dynamics that were learned in the initial training run. Subsequently, a second training run was performed (with the generator's dynamics locked) in which all trials were included (including the held-out trials, i.e., the trials from the previous held-out conditions). This allowed the initial conditions encoder RNN to learn a mapping from the new trials to initial conditions for the generator RNN, but did not allow the generator to learn any new dynamics from the held-out trials.

5.3 Figure 4 - pKinematic predictions of LFADS multi-session and single-session models

We used optimal linear estimation to create decoders to predict x and y reaching velocities. For decoding from LFADS, we used the factors rather than the predicted firing rates, as the neurons recorded on an individual session could unevenly represent the full set of reaching directions well, even if the underlying factors from which the rates are extracted represent all directions evenly. For single-dataset LFADS models, we fit individual decoders to map from each model's factors to x and y velocities. For the stitched multi-session LFADS model, a single decoder was fit and cross-validated on all datasets simultaneously. We then computed the goodness of fit (R^2) and averaged across x and y velocities. Aside from decoding from LFADS factors rather than LFADS rates, the inputs to the decoder were prepared and the cross-validated decoding performance evaluated as described in *5.1*. For Gaussian smoothing, the millisecond-binned spike trains were convolved with a Gaussian function with standard deviation (s.d.) of 40 ms. For GPFA, we swept the spike bin width and the number of latent factors to determine the optimal hyperparameters for decoding, which were 20 ms bins and 20 latent factors for these datasets. In all cases, to decode kinematics, the neural features were 'lagged' by 90 ms to account for delays between neural activity and measured kinematics, and the neural features and kinematics were resampled at 20 ms.

For reaction time prediction, we used a largely unsupervised method previously described in [21]. Briefly, for each of the single-session models and the multi-session model, we performed demixed principal components analysis (dPCA[6]) on the factor outputs. We then projected the factors along the highest-variance, condition-independent mode, and normalized the projection to a range of 0 to 1. This projection of the data we refer to as the condition independent signal (CIS), following [21]. We then took the time at which the CIS crossed a certain threshold on each trial to be the predicted reaction time, and computed the correlation coefficient between predicted and actual reaction times. For each model, we then optimized only the threshold to maximize the correlation coefficient between time of threshold crossing and reaction time, though the results were not sensitive to the choice of threshold.

Factor trajectories were trial-averaged for each reaching direction and each dataset. With these trial-averaged factor trajectories, we used dPCA to identify the CIS dimension, as well as 8 dimensions which preferentially explained condition-dependent variance (variation due to reach direction, and mixtures of reach direction and time). In this 8-dimensional condition-dependent space, we used jPCA to find a plane where trajectories exhibited rotational structure[3]. We then constructed a 3-dimensional subspace for visualization by taking the CIS dimension as well as the two dimensions comprising the first jPCA plane.

5.4 Figure 5 - tSNE visualization for CursorJump data

The pattern of inputs inferred by LFADS for individual trials were mapped into a 2-dimensional space using t-SNE. Data were aligned to the time of perturbation for perturbed trials or the mean perturbation time for the given target direction for unperturbed trials (407 ms for downward targets, 487 ms for upward targets). t-SNE was performed using the t-SNE toolbox for MatLab (https://lvdmaaten.github.io/tsne/). Inferred inputs were calculated via posterior averaging, as described in section 1.11. LFADS inferred the input values at 10 ms resolution (i.e., the resolution at which the neural data was binned before being passed into LFADS). These values were then smoothed using a causal Gaussian filter with a 20 ms standard deviation. Data fed into t-SNE consisted of the inferred input values from 40 ms to 240 ms after the time at which the task perturbation occurred (or after the mean perturbation time for unperturbed trials, as described above). t-SNE initially pre-processes data by reducing its dimensionality via PCA, and the dimensionality of the pre-processed data was chosen to be 30 dimensions. The t-SNE perplexity parameter was set to 30, and sweeping this parameter between 10 to 50 had little qualitative effect on the discernibility of the three data clusters.

5.5 Figure 6 - LFP analysis

For both human (participant T7) and monkey (J) data, recorded LFP was originally sampled with high bandwidth (human: 30 kHz, monkey: 2kHz). Human data was digitally re-referenced using common-average referencing to remove global noise artifacts. Human and monkey data were low-pass filtered with a 75 Hz cutoff frequency using a 4th order Butterworth filter to minimize the contribution of action potentials to the LFP signal. Both a forwards and backwards pass of the filter (i.e., acausal filtering) were used to minimize group delay. Data were then filtered again with an anti-aliasing filter (8th order Chebyshev Type I lowpass filter with cutoff of 0.8 * sampling frequency / 2) and then resampled to 1 kHz for all subsequent analyses. Data analyzed were from a center-out-and-back movement paradigm. Participant T7 made movements of his index finger on a touchpad to control a cursor's on-screen movements. Monkey J made movements of his hand in free space to control the movements of a cursor. Data analyzed were from the first 300 ms (participant T7) or 250 ms (monkey J) after target onset. For each recording channel on the electrode arrays, cross-correlograms were computed between the measured spiking activity and the recorded local field potentials on the same electrode, on a single trial basis. Cross-correlograms were then averaged across all trials. For the shuffle analyses, spiking data from an individual trial was cross-correlated with LFP data from a random trial, and these correlograms were averaged across trials.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

We would like to thank John P. Cunningham and Jascha Sohl-Dickstein for extensive conversation. We thank Mark M. Churchland for contributions to data collection for monkey J, Christine Blabe and Paul Nuyujukian for assistance with research sessions with participant T5, Emad Eskandar for array implantation with participant T7, and Brittany Sorice and Anish Sarma for assistance with research sessions with participant T7. R.J. participated in this work while at Google, Inc. L.F.A.'s research was supported by US National Institutes of Health grant MH093338, the Gatsby Charitable Foundation through the Gatsby Initiative in Brain Circuitry at Columbia University, the Simons Foundation, the Swartz Foundation, the Harold and Leila Y. Mathers Foundation, and the Kavli Institute for Brain Science at Columbia University. C.P. was supported by a postdoctoral fellowship from the Craig H. Neilsen Foundation for spinal cord injury research and the Stanford Dean's Fellowship. S.D.S. was supported by the ALS Association's Milton Safenowitz Postdoctoral Fellowship. K.V.S.'s research was supported by the following awards: an NIH-NINDS award (T-R01NS076460), an NIH-NIMH award (T-R01MH09964703), an NIH Director's Pioneer award (8DP1HD075623), a DARPA-DSO 'REPAIR' award (N66001–10-C-2010), a DARPA-BTO 'NeuroFAST' award (W911NF-14–2-0013), a Simons Foundation Collaboration on the Global Brain award (325380), and the Howard Hughes Medical Institute. J.M.H.'s research was supported by NIH-NIDCD R01DC014034. K.V.S. and J.M.H.'s research was supported by Stanford BioX-NeuroVentures, Stanford Institute for Neuro-Innovation and Translational Neuroscience, Garlick Foundation and Reeve Foundation. L.R.H's research was supported by NIH-NIDCD R01DC009899, Rehabilitation Research and Development Service, Department of Veterans Affairs (B6453R), MGH-Deane Institute for Integrated Research on Atrial Fibrillation and Stroke; Executive Committee on Research, Massachusetts General Hospital.

The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health, the Department of Veterans Affairs, or the United States Government. BrainGate CAUTION: Investigational Device. Limited by Federal Law to Investigational Use.

Competing Financial Interests Statement

The authors declare no competing financial interests. J.M.H is on the Medical Advisory Boards of Enspire DBS and Circuit Therapeutics, and the Surgical Advisory Board for Neuropace, Inc. K.V.S. is a consultant to Neuralink Corp. and on the Scientific Advisory Boards of CTRL-Labs, Inc. and Heal, Inc. These entities did not support this work.

References

1. Afshar A et al. Single-trial neural correlates of arm movement preparation. Neuron 71, 555–564 (2011). [PubMed: 21835350]
2. 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). [PubMed: 25959731]
3. Churchland MM et al. Neural population dynamics during reaching. Nature (2012). doi:10.1038/nature11129
4. Harvey CD, Coen P & Tank DW Choice-specific sequences in parietal cortex during a virtual-navigation decision task. Nature 484, 62–68 (2012). [PubMed: 22419153]
5. Kaufman MT, Churchland MM, Ryu SI & Shenoy KV Cortical activity in the null space: permitting preparation without movement. Nat. Neurosci 17, 440–448 (2014). [PubMed: 24487233]
6. Kobak D et al. Demixed principal component analysis of neural population data. Elife 5, (2016).
7. Mante V, Sussillo D, Shenoy KV & Newsome WT Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature 503, 78–84 (2013). [PubMed: 24201281]
8. Pandarinath C et al. Neural population dynamics in human motor cortex during movements in people with ALS. Elife 4, (2015).
9. Sadtler PT et al. Neural constraints on learning. Nature in press, 423–426 (2014).
10. Shenoy KV, Sahani M & Churchland MM Cortical control of arm movements: a dynamical systems perspective. Annu. Rev. Neurosci 36, 337–359 (2013). [PubMed: 23725001]

11. Ahrens MB et al. Brain-wide neuronal dynamics during motor adaptation in zebrafish. Nature 485, 471–477 (2012). [PubMed: 22622571]
12. Yu BM et al. Gaussian-Process Factor Analysis for Low-Dimensional Single-Trial Analysis of Neural Population Activity. J. Neurophysiol 102, 614–635 (2009). [PubMed: 19357332]
13. Zhao Y & Park IM Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains. Neural Comput. 29, 1293–1316 (2017). [PubMed: 28333587]
14. Aghagolzadeh M & Truccolo W Latent state-space models for neural decoding. Conf. Proc. IEEE Eng. Med. Biol. Soc 2014, 3033–3036 (2014). [PubMed: 25570630]
15. Gao Y, Archer EW, Paninski L & Cunningham JP in Advances in Neural Information Processing Systems 29 (eds. Lee DD, Sugiyama M, Luxburg UV, 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>
16. Kao JC et al. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. Nat. Commun 6, (2015).
17. Macke JH et al. Empirical models of spiking in neural populations. Advances in neural information processing systems 1350–1358 (2011).
18. 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>
19. Petreska B et al. in Advances in Neural Information Processing Systems 24 (eds. Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F & Weinberger KQ) 756–764 (Curran Associates, Inc., 2011). at <http://papers.nips.cc/paper/4257-dynamical-segmentation-of-single-trials-from-population-neural-data.pdf>
20. Kato S et al. Global brain dynamics embed the motor command sequence of Caenorhabditis elegans. Cell 163, 656–669 (2015). [PubMed: 26478179]
21. Kaufman MT et al. The largest response component in motor cortex reflects movement timing but not movement type. eNeuro 3, ENEURO.0085–16.2016 (2016).
22. Gao P & Ganguli S On simplicity and complexity in the brave new world of large-scale neuroscience. Curr. Opin. Neurobiol 32, 148–155 (2015). [PubMed: 25932978]
23. Kingma DP & Welling M Auto-Encoding Variational Bayes. arXiv [stat.ML] (2013). at <http://arxiv.org/abs/1312.6114v10LB-BIpxX>
24. Doersch C Tutorial on Variational Autoencoders. arXiv [stat.ML] (2016). at <http://arxiv.org/abs/1606.05908LB-XZTq>
25. Sussillo D, Jozefowicz R, Abbott LF & Pandarinath C LFADS - Latent Factor Analysis via Dynamical Systems. arXiv (2016). at <http://arxiv.org/abs/1608.06315>
26. Salinas E & Abbott LF Vector reconstruction from firing rates. J. Comput. Neurosci 1, 89–107 (1994). [PubMed: 8792227]
27. Willett FR et al. Feedback control policies employed by people using intracortical brain-computer interfaces. J. Neural Eng 14, (2017).
28. Turaga S et al. in Advances in Neural Information Processing Systems 26 (eds. Burges CJC, Bottou L, Welling, Ghahramani Z & Weinberger KQ) 539–547 (Curran Associates, Inc., 2013). at <http://papers.nips.cc/paper/4874-inferring-neural-population-dynamics-from-multiple-partial-recordings-of-the-same-neural-circuit.pdf>
29. Nonnenmacher M, Turaga SC & Macke JH in Advances in Neural Information Processing Systems 30 (eds. Guyon I et al.) 5706–5716 (Curran Associates, Inc., 2017). at <http://papers.nips.cc/paper/7153-extracting-low-dimensional-dynamics-from-multiple-large-scale-neural-population-recordings-by-learning-to-predict-correlations.pdf>
30. Donoghue JP, Sanes JN, Hatsopoulos NG & Gaal G Neural discharge and local field potential oscillations in primate motor cortex during voluntary movements. J Neurophysiol 79, 159–173 (1998). [PubMed: 9425187]
31. Murthy VN & Fetz EE Synchronization of neurons during local field potential oscillations in sensorimotor cortex of awake monkeys. J. Neurophysiol 76, 3968–3982 (1996). [PubMed: 8985893]

32. Fries P A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. Trends Cogn. Sci 9, 474–480 (2005). [PubMed: 16150631]
33. Yuste R From the neuron doctrine to neural networks. Nat. Rev. Neurosci 16, (2015).
34. Gilja V et al. Clinical translation of a high-performance neural prosthesis. Nat. Med 21, (2015).
35. Pandarinath C et al. High performance communication by people with paralysis using an intracortical brain-computer interface. Elife 6, (2017).
36. Sussillo D et al. A recurrent neural network for closed-loop intracortical brain-machine interface decoders. J. Neural Eng 9, 26027 (2012).
37. Sussillo D, Stavisky SD, Kao JC, Ryu SI & Shenoy KV Making brain–machine interfaces robust to future neural variability. Nat. Commun 7, 13749 (2016). [PubMed: 27958268]
38. Ezzyat Y et al. Closed-loop stimulation of temporal cortex rescues functional networks and improves memory. Nat. Commun 9, 365 (2018). [PubMed: 29410414]
39. Klinger NV & Mittal S Clinical efficacy of deep brain stimulation for the treatment of medically refractory epilepsy. Clin. Neurol. Neurosurg 140, 11–25 (2016). [PubMed: 26615464]
40. Little S et al. Adaptive deep brain stimulation in advanced Parkinson disease. Ann. Neurol 449–457 (2013). doi:10.1002/ana.23951
41. Rosin B et al. Closed-loop deep brain stimulation is superior in ameliorating parkinsonism. Neuron 72, 370–384 (2011). [PubMed: 22017994]
42. Williamson RS, Sahani M & Pillow JW The equivalence of information-theoretic and likelihood-based methods for neural dimensionality reduction. PLoS Comput. Biol 11, e1004141 (2015). [PubMed: 25831448]

Methods-only References

43. Rezende DJ, Mohamed S & Wierstra D Stochastic backpropagation and approximate inference in deep generative models. in International Conference on Machine Learning, 2014 (2014).
44. Gregor K, Danihelka I, Graves A, Rezende DJ & Wierstra D DRAW: A Recurrent Neural Network For Image Generation. arXiv [cs.CV] (2015). at <http://arxiv.org/abs/1502.04623LB-VErwh>
45. Krishnan RG, Shalit U & Sontag D Deep Kalman Filters. arXiv Prepr. arXiv1511.05121 (2015).
46. Chung J, Gulcehre C, Cho K & Bengio Y Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv Prepr. arXiv1412.3555 (2014).
47. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I & Salakhutdinov RR Improving neural networks by preventing co-adaptation of feature detectors. arXiv Prepr. arXiv1207.0580 (2012).
48. Zaremba W, Sutskever I & Vinyals O Recurrent neural network regularization. arXiv Prepr. arXiv1409.2329 (2014).
49. Bowman SR et al. Generating sentences from a continuous space. Conf. Comput. Nat. Lang. Learn (2016).
50. Sussillo D & Abbott LF Generating coherent patterns of activity from chaotic neural networks. Neuron 63, 544–557 (2009). [PubMed: 19709635]
51. 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). [PubMed: 26075643]
52. Rajan K, Harvey CD & Tank DW Recurrent Network Models of Sequence Generation and Memory. Neuron 90, 1–15 (2016). [PubMed: 27054611]
53. Chung J et al. A Recurrent Latent Variable Model for Sequential Data. in Advances in Neural Information Processing Systems (NIPS) (2015).
54. Gao Y, Buesing L, Shenoy KV & Cunningham JP 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>
55. Bayer J & Osendorfer C Learning stochastic recurrent networks. arXiv Prepr. arXiv1411.7610 (2014).

56. 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).

57. 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).

58. Gilja V et al. A high-performance neural prosthesis enabled by control algorithm design. Nat. Neurosci 15, 1752–7 (2012). [PubMed: 23160043]

59. Maaten L van der & Hinton G Visualizing data using t-SNE. J. Mach. Learn. Res 9, 2579–2605 (2008).

60. Fan JM et al. Intention estimation in brain--machine interfaces. J. Neural Eng 11, 16004 (2014).

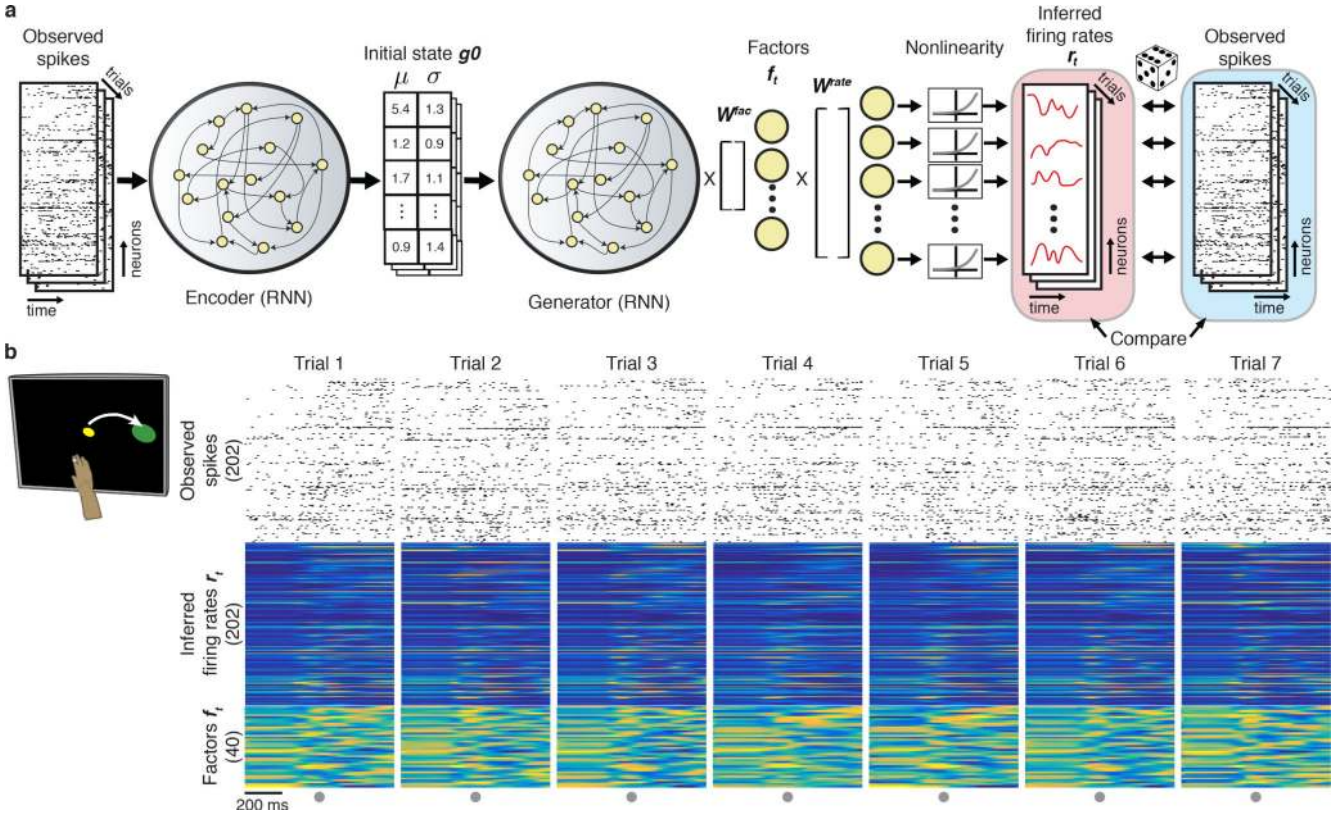

Figure 1.

LFADS is a generative model that assumes that observed single-trial spiking activity is generated by an underlying dynamical system. (**a**) LFADS takes a given recording (far left), reduces it to a latent code consisting of an inferred initial condition (middle), and then attempts to infer rates that are consistent with the observed data (right, pink panel) from that latent code. I.e. LFADS auto-encodes the trial via a sequential auto-encoder. Working from right to left in the panel, for the i^{th} neuron, LFADS infers rates at time t, $r_{t,i}$, for each of 202 channels, and the observed spike counts (blue panel) are assumed to be Poisson distributed count observations of these underlying rates. The likelihood of the observed spikes given the inferred rates serves as the cost function used to optimize the weights of the model. The rates are linear readouts from a set of low-dimensional factors $\boldsymbol{f_t}$ (40 in this example) via a readout matrix $\boldsymbol{W^{rate}}$. The factors are defined as linear readouts from a dynamical generator (an RNN), via a readout matrix $\boldsymbol{W^{fac}}$. Activity of the generator is determined by its per-trial component, the initial condition ($\boldsymbol{g0}$), and its recurrent connectivity, which is fixed for all trials. The initial condition $\boldsymbol{g0}$ is determined for individual trials via an encoder RNN. (**b**) Example spiking activity recorded from M1/PMd as a monkey performed a reaching task, as well as the corresponding rates $\boldsymbol{r_t}$ and factors $\boldsymbol{f_t}$ inferred by LFADS (7 example trials are shown). Circles denote time of movement onset.

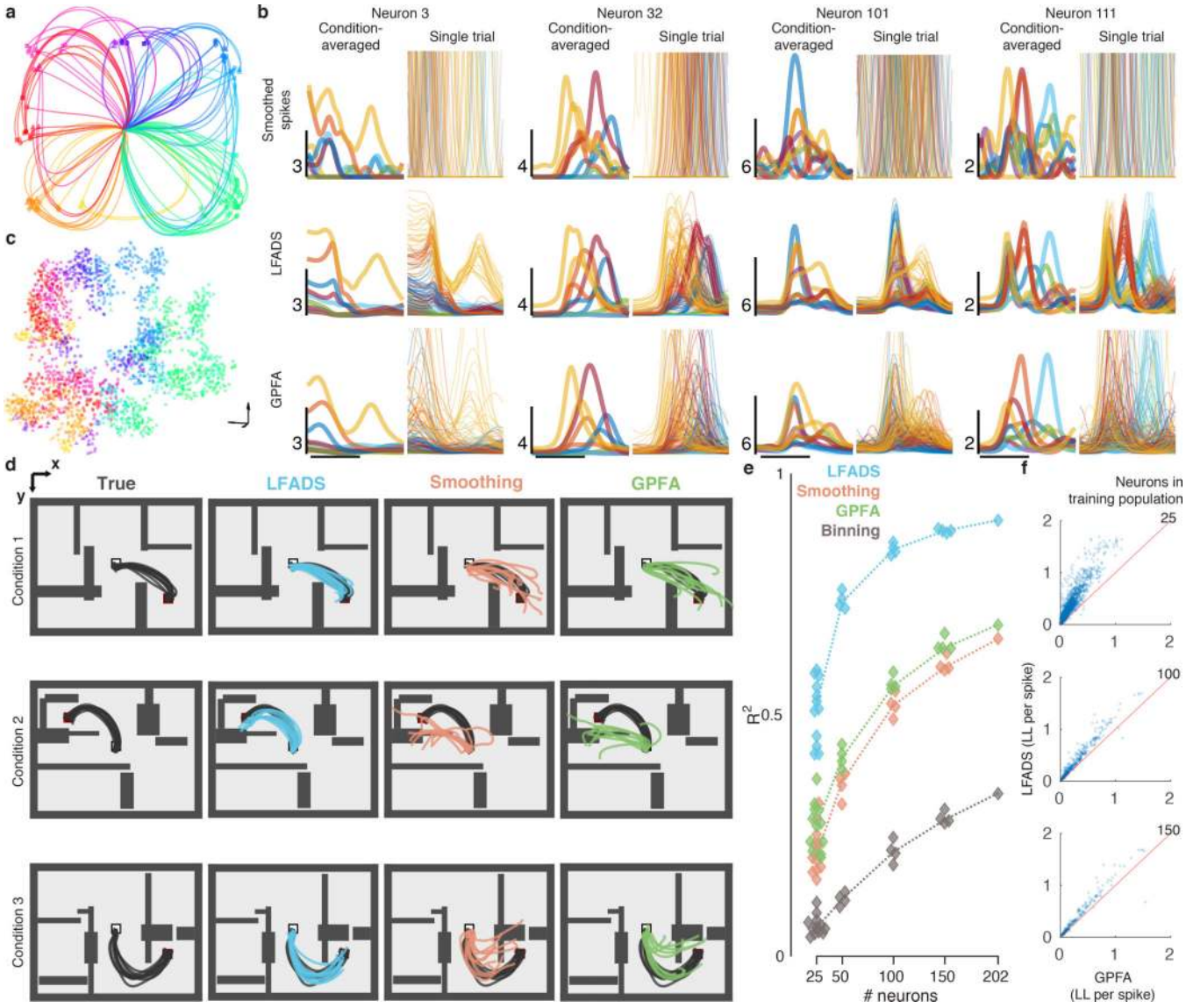

Figure 2.

Application of LFADS to a "Maze" reaching task. (**a**) A monkey was trained to perform arm reaching movements to guide a cursor in a 2-D plane from a starting location (center of the workspace) to peripheral targets. Individual reaches are colored by target location. Virtual barriers in the workspace facilitated instruction of curved (or straight) reaches on a per-condition basis (see (**d**) for examples). (**b**) Comparison of condition-averaged (left) and single-trial (right) rates for 4 individual neurons (columns) for three different methods (rows). *Left:* Each trace represents a different reach condition (8 selected of 108 total). *Right*: Each trace represents an individual trial (same color scheme as the condition-averaged panels). *Top row*: PSTHs created by smoothing observed spikes with a Gaussian kernel (30 ms s.d.). *Middle row*: LFADS-inferred rates. *Bottom row*: GPFA-inferred firing rates, created by fitting a generalized linear model (GLM) to map the GPFA-inferred factor representations onto the true spiking activity. Horizontal scale bar represents 300 ms. Vertical scale bar denotes rate (spikes/sec). PSTHs for all neurons are shown in Supp. Data

1. (**c**) Application of t-SNE to the generator initial conditions (***g0***). Each point represents the reduction of the ***g0*** vector into a 3-D t-SNE space for an individual trial (2296 trials total), 2-D projection shown, full 3-D projection shown in Supp. Video 1. Trials are color coded by the angle of the reach target [same as (a)]. (**d**) Decoding reaching kinematics using optimal linear estimation. Each row shows an example condition (3 shown, of 108 total). Column 1: true reach trajectories (black traces, 10 example trials per condition). Columns 2–4: examples of cross-validated reconstruction of these trajectories using OLE applied to the neural data, which was first de-noised either via LFADS, by smoothing with a Gaussian filter (40 ms s.d.), or using GPFA to reduce its dimensionality. (**e**) Decoding accuracy was quantified by measuring variance explained (R^2) between the true and decoded velocities for individual trials across the entire dataset (2296 trials), for all three techniques and additionally for simple binning of the neural data. Accuracy was also measured for random sub-samples from the full neural population of 202 neurons. Dotted lines connect the median R^2 values for each population size. (**f**) LFADS-inferred factors are informative about neurons that are held-out from model training. LFADS models and GPFA were fit to subsets of the full population of the 202 neurons [same populations as in (e)]. We then used a GLM to map the latent state estimates produced by LFADS or GPFA onto the binned spike counts (20 ms bins) for the remaining held-out neurons, e.g., for a model trained with 25 neurons, there are 177=202–25 held out neurons. We evaluated the cross-validated performance of the fit GLM models using log likelihood (LL) per spike[42]. Each point represents a given held-out neuron for a given random sampling of the population.

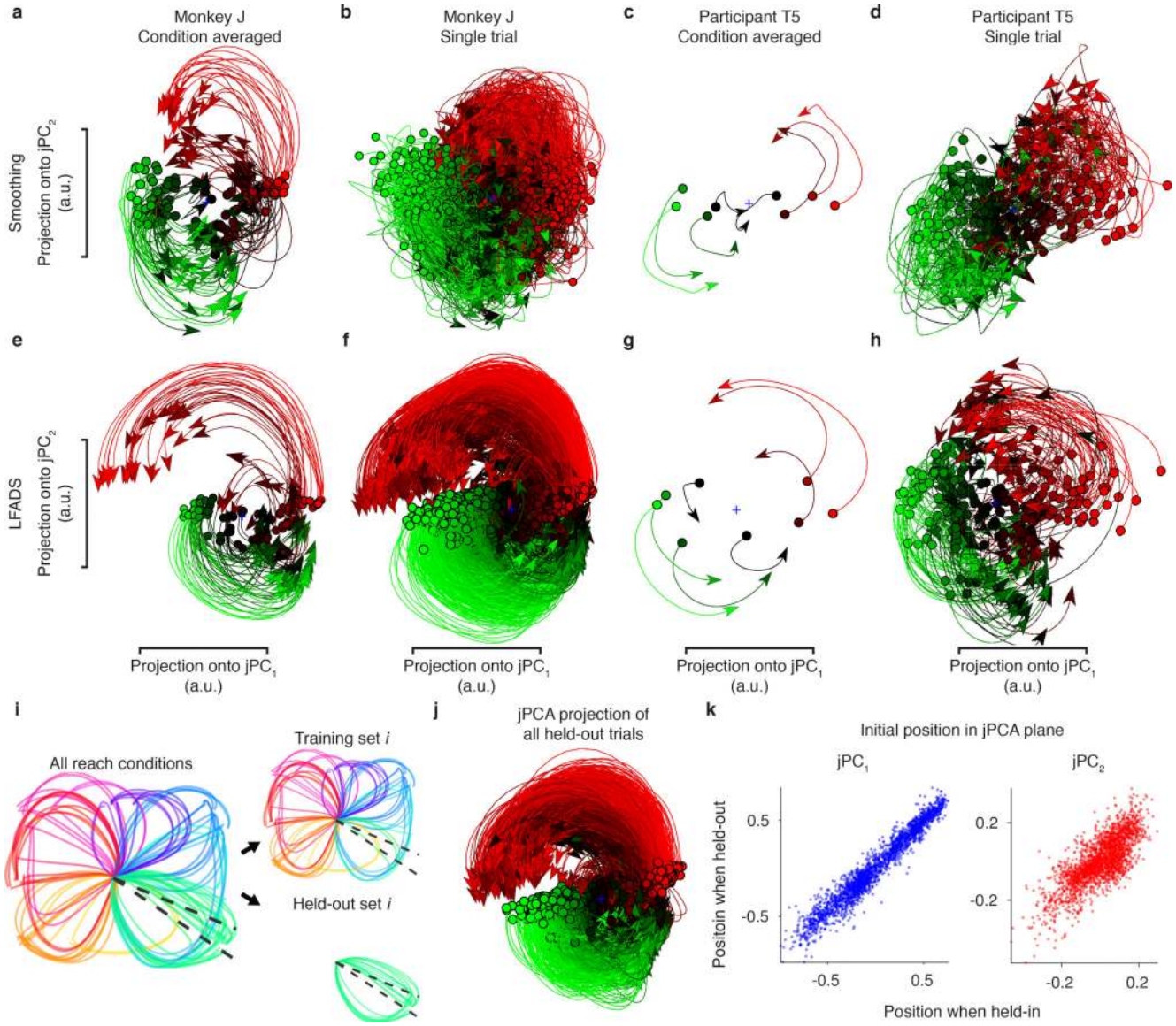

Figure 3.

LFADS uncovers known rotational dynamics in monkey and human motor cortical activity on a single-trial basis. (**a, c**) Rotational dynamics underlying the neural population state accompany the transition between pre- and peri-movement activity, and have been previously described for monkey[3] and human[8] motor cortical activity by projecting condition-averaged activity into a low-dimensional plane using jPCA. Each trace shows the neural population state trajectories for a single task condition (monkey: 108 reaching conditions; human: 8 intended movement directions). (**b, d**) When the same low-dimensional projection is applied to the single-trial data, dynamics are less clear due to the inherent noise of single-trial neural population activity. (**e, g**) When LFADS is applied, the condition-averaged inferred rates exhibit similar underlying dynamic structure for monkey and human. (**f, h**) Additionally, the same dynamic structure is now clearly present on individual trials (monkey: 2296 trials; human: 114 trials). (**i-k**) Testing generalizability of

the generator's dynamics to held out conditions. (**i**) Conditions were binned by the angle of the reach target (black dashed lines), resulting in 19 sets. 19 LFADS models' generator dynamics were then trained, each on 18 subsets of the data with 1 subset held out, and then evaluated on the held-out subset. (**j**) LFADS-inferred rates for held-out conditions were combined across the 19 models and were projected into the jPCA space found by training an LFADS model on all conditions (i.e., panel f). (**k**) Correspondence between initial position in jPCA space when a trial is used in the training set for an LFADS model and when it is held-out (Pearson's correlation coefficient $r = 0.97, 0.77$ for jPC_1, jPC_2 respectively). Each dot represents an individual trial (2296 trials).

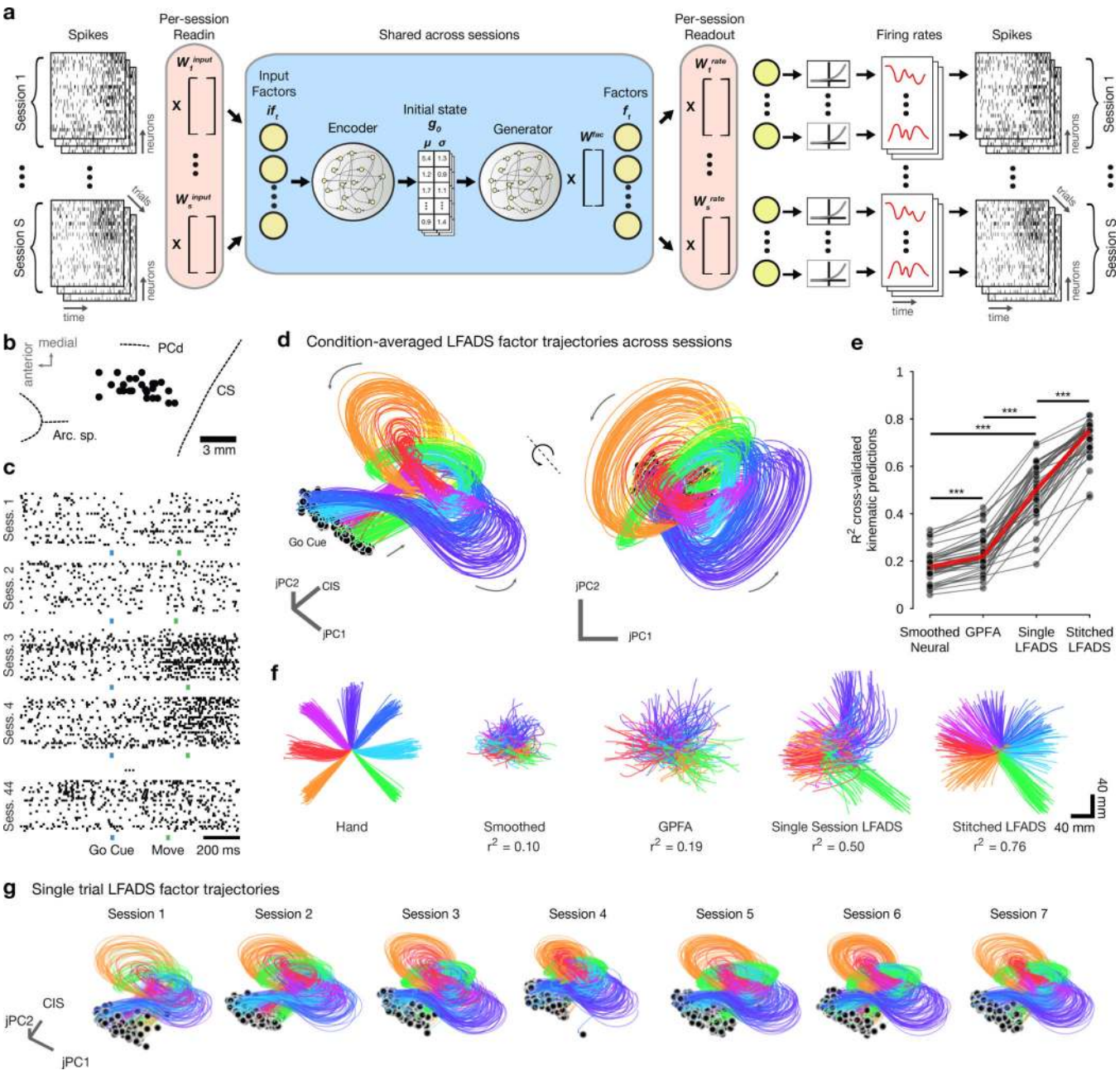

Figure 4.

Using "dynamic neural stitching," LFADS combines data from separately collected, non-overlapping recordings of the neural population by learning one consistent dynamical model. (**a**) Schematic of the LFADS architecture adapted for dynamic neural stitching. Per-session "readin" matrices $\boldsymbol{W}_s^{input}$ and "readout" matrices $\boldsymbol{W}_s^{rate}$ are used to map from each dataset's rates to the input factors and from the factors back out to rates, respectively (pink areas). The encoder RNN, generator RNN, and factor readout matrix $\boldsymbol{W}^{fac}$ are shared among datasets (blue area). For this example, each $\boldsymbol{W}_s^{rate}$ was learned whereas $\boldsymbol{W}_s^{input}$ was set using a principal components regression approach (see Online Methods). A total of 44 individual recording sessions using 24 channel linear multielectrode arrays were used. (**b**) Locations of

linear electrode array penetrations in the precentral gyrus from which each dataset was collected. Dashed lines indicate approximate locations of nearby sulcal features based on stereotaxic locations. Arc. Sp.: arcuate spur, PCd: precentral dimple, CS: central sulcus. (**c**) Example single-trial rasters for nearly identical upwards reaches performed on a subset of 5 of the 44 recording sessions. Each raster has 24 rows corresponding to the 24 channels of the linear array, but the neurons recorded on each session are entirely distinct from each other. (**d**) After training, the multi-session stitched LFADS model produced consistent factor trajectories for each behavioral condition across recording sessions. Traces are condition-averaged factor trajectories for the multi-session stitched LFADS model projected into a subspace which spans the condition independent signal (CIS) and the first jPCA plane (see Methods). LFADS factors are averaged over all trials in each reach direction for each recording session and projected into this subspace to produce a single trajectory; the color of each trajectory represents the reach direction. The spatial proximity of the trajectories for a given direction across the sessions (44 trajectories of each color) illustrates the consistency of the representation across sessions. (**e**) R^2 values between arm kinematics and either smoothing neural data, GPFA, single-session or stitched LFADS factor decodes. A single shared decoder was fit for the stitched model; a separate decoder was fit for each single-session model. "***" indicates significant improvement in median R^2, $p < 10^{-8}$, Wilcoxon signed-rank test. (**f**) Actual recorded hand position traces for center out reaching task (left), alongside kinematic decodes for a representative single session (session 32), for smoothed neural data, GPFA, single-session LFADS, and stitched LFADS (left to right). Colors indicate reach direction. (**g**) Single-trial factor trajectories from the stitched LFADS model. Only the first seven of 44 sessions are shown for ease of presentation (see also Supp. Video 3).

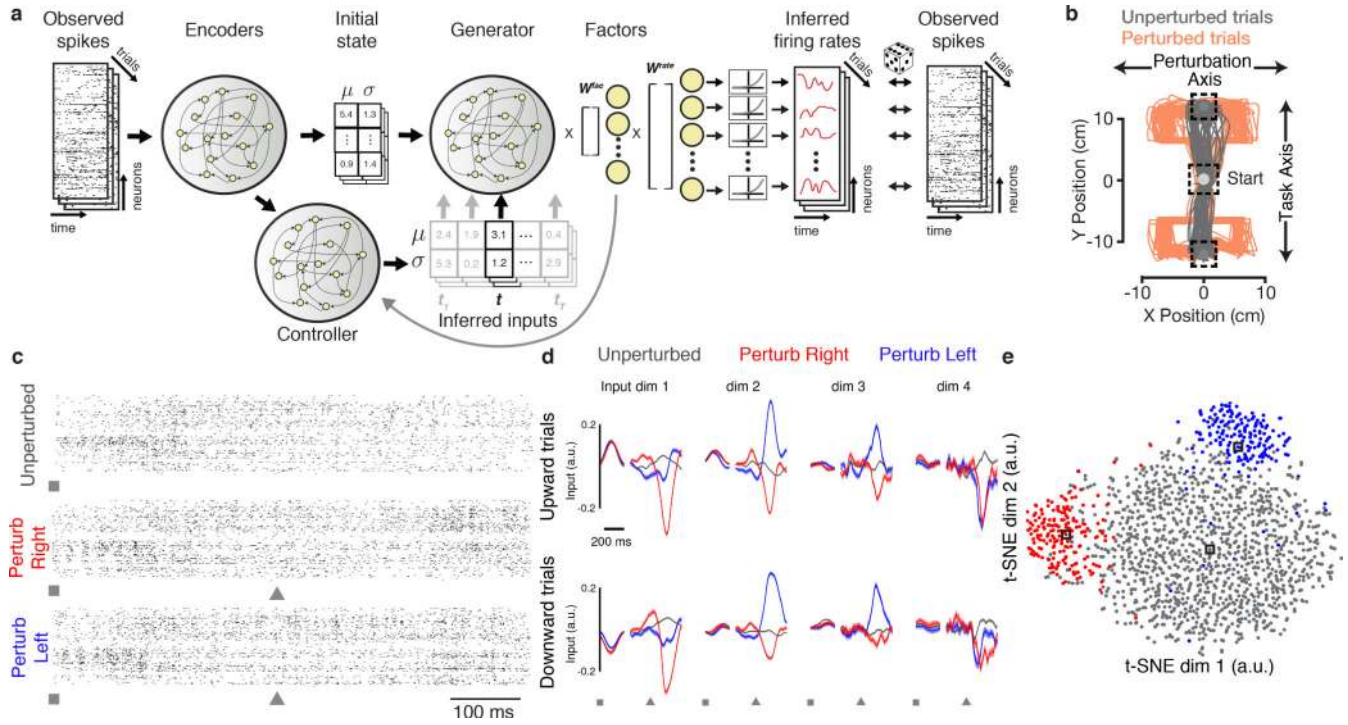

Figure 5.

LFADS uncovers the presence, identity and timing of unexpected perturbations in the "Cursor Jump" task. (**a**) Schematic of the LFADS architecture adapted for inferring inputs to a neural population. As before, LFADS reduces individual trials to the initial state of the generator RNN (***g0***). However, now the activity of the generator is additionally determined by a set of time-varying inferred inputs ($\boldsymbol{u_t}$), modeled stochastically like ***g0*** with a mean and variance, which are inputted to the generator at each time point. The inferred input $\boldsymbol{u_t}$ is output by a controller RNN, which receives time-varying input from the encoding network, as well as the factors representation at the preceding timestep. (**b**) Schematic depicting the "Cursor Jump" task. The position of a monkey's hand was linked to the position of an on-screen cursor, and the monkey made reaching movements to steer the cursor toward upward or downward targets. In unperturbed trials (grey traces), the monkey made straight reaches to the target. In perturbed trials (orange traces), the cursor's position was offset to the left or right during the course of the reaching movement, and the monkey made corrective movements to acquire the target. (**c**) Spiking activity from M1/PMd arrays during three example reach trials to downward targets for the unperturbed (top), perturb right (middle), and perturb left (bottom) conditions. Squares denote time of target onset, and triangles denote the time of an unexpected perturbation. (**d**) LFADS was allowed 4 inferred inputs to model the neural activity. For presentation, two trial alignments were used prior to averaging: the initial portion of the trials was aligned to the time of target onset, while the latter portion of the trials was aligned by perturbation time (or, for unperturbed trials, the time at which a perturbation would have occurred based on the cursor's trajectory). The gap in the traces denotes the break in alignment. Inferred input values were averaged across trials for upward (top) and downward (bottom) trials (mean ± s.e.m. is shown, grey: unperturbed trials, blue: perturb left trials, red: perturb right trials). Around the time of target onset, the

identity of the target (up vs. down) is modeled by the inputs (e.g., dimension 1). Around the time of the perturbation, LFADS used specific inferred input patterns to model each perturbation type (e.g., dimensions 1 & 2). Input traces were smoothed with a causal Gaussian filter (20 ms s.d.). (**e**) The single-trial input patterns around the time of perturbation (all downward trials) were projected into a low-dimensional space using t-SNE and colored by the three perturbation types (unperturbed, left perturbation, right perturbation). Black boxes denote locations in t-SNE space for the example trials shown in panel **c**.

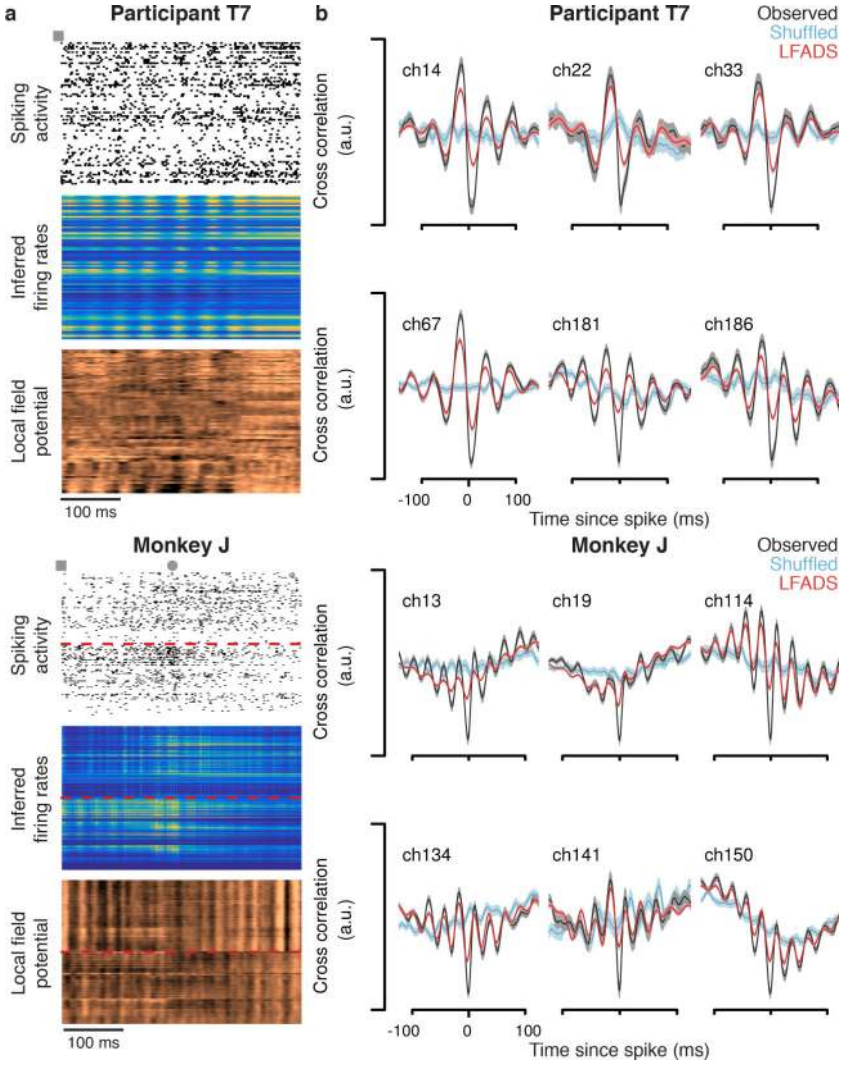

Figure 6.

When inferred inputs are allowed (Fig. 5a), LFADS uncovers fast oscillatory structure in neural firing patterns. (**a**) Example single-trial spiking activity recorded from human M1 and monkey M1/PMd, as well as LFADS-inferred rates, and local field potentials. 400 ms of data are shown, beginning at the time of target presentation during an 8-target center-out-and-back movement paradigm. For T7, analyses were restricted to channels that showed significant modulation during movement attempts (78/192 channels). Dashed red lines overlaid on monkey data segregate the M1 array (upper halves) and PMd array (lower halves). Squares denote time of target onset. For Monkey J, where movement was measurable, circle denotes time of movement onset. (**b**) Cross-correlations between the local field potentials recorded on each electrode and the observed spiking activity (black traces; mean ± s.e.m.) or the LFADS-inferred rates (red traces) for several example channels (participant T7: 142 trials; monkey J: 373 trials). LFP were first low-pass filtered (75 Hz cutoff frequency). Randomly shuffling the trial identity (i.e., correlating spikes from one trial

with LFP from another) largely removed the fast, oscillatory components in the cross-correlograms (blue traces).

Methods 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	N	F	$\lvert u_t \rvert$	g0 E dim	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.

Methods Table 2.

Signal collection technology and spike detection methods.

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