nature methods

ARTICLES

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

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

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

Neuroscience is experiencing a revolution in which simultaneous recording of thousands of neurons is revealing population dynamics that are not apparent from single-neuron responses. This structure is typically extracted from data averaged across many trials, but deeper understanding requires studying phenomena detected in single trials, 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, a deep learning method to infer latent dynamics from single-trial neural spiking data. When applied to a variety of macaque and human motor cortical datasets, latent factor analysis via dynamical systems accurately predicts observed behavioral variables, extracts precise firing rate 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.

In many brain areas, the activity of large populations of neurons is often well described by low-dimensional dynamics[1–9]. Thus, one may begin to understand the computations of brain areas without observing all of their neurons because these computations can be described by the dynamics of a modest number of underlying 'latent factors'[10], where each factor captures a pattern of co-activation across neurons. 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 (BMIs). However, recovering population dynamics on single trials is difficult due to trial-to-trial variability 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 machine-learning method based on non-linear artificial recurrent neural networks (RNNs), termed latent factor analysis via dynamical systems (LFADS). LFADS is based on the idea that neural data can be generated by a dynamical system, which is defined by equation (1). LFADS models the following generic dynamical system:

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t),\ \mathbf{u}(t)) \tag{1}$$

The state of the dynamical system $\mathbf{x}(t)$ is updated by the vector-valued function $\mathbf{F}$, which is non-linear and potentially complicated, accepts optional input $\mathbf{u}(t)$, and is seeded by an initial condition, $\mathbf{x}(0)$. LFADS models $\mathbf{F}$, $\mathbf{x}(0)$, and optionally $\mathbf{u}(t)$. By modeling equation (1), LFADS assumes that the process that produces the observed spiking activity can be modeled as a dynamical system. The optional input is constrained to be considerably less dynamically complex than $\mathbf{x}(t)$. 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 (that is, 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

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

inputs; and (4) spiking variability distributed according to a Poisson distribution.

We describe a concrete implementation that takes observed neural data as an input and estimates the data's latent neural state, initial conditions, inputs, and de-noised firing rates (rates). In LFADS, an RNN (the 'generator') produces the underlying dynamics (assumption 1). We assume that a continuous valued dynamical system can describe the dynamics of neural data. LFADS extracts dynamic 'factors' from this system and uses them to infer rates for the recorded neurons. We model observed action potentials as samples from an inhomogenous Poisson process whose rate corresponds to the inferred firing rate for the given neuron (assumption 4). Additional RNNs (the 'encoder' and 'controller') extract initial conditions and input for the generator (assumptions 2 and 3) from the observed spiking data for each trial. Yet, beyond binned spike sequences, no other trial-specific information such as condition or behavioral information is supplied.

A strength of this approach is that non-linear RNNs can reproduce the complex temporal activity patterns of neural data. In addition, LFADS can find low-dimensional dynamics that explain the recorded data by constraining the number of factors in the model. This is consistent with observations that the dimensionality of neural population activity in areas such as motor and prefrontal cortices is, in many cases, much lower than the number of recorded neurons[3,7,20–22].

Here, we apply LFADS to a variety of datasets from rhesus macaque motor (M1) and dorsal premotor (PMd) cortices, as well as human M1 (macaque data were previously recorded at Stanford University). We show that rates extracted by LFADS estimate behavioral variables 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 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 five months of recording, to improve its performance on the individual trials from each recording session. Finally, we demonstrate that LFADS can 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 rates associated with local field potentials (LFPs).

Results

Overview of LFADS. To introduce LFADS (Figs 1–4 and Supplementary Figs 1–6), we start with a simplified dynamical systems model that ignores the input in equation (1), yielding

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) \quad (2)$$

Beyond Poisson spiking variability, all trial-to-trial variability in this system is captured by the initial condition, $\mathbf{x}(0)$, for that trial.

LFADS is a sequential adaptation of a variational auto-encoder (VAE)[23–25] that maximizes 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 the Methods, and associated source code is available as Supplementary Software and at https://lfads.github.io).

Working from output to input, LFADS models the single-trial spiking observations at time t as stochastic (Poisson) spike counts generated from a vector of underlying firing rates $\mathbf{r}_t$ (Fig. 1a). 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 $\mathbf{f}_t$ by a readout matrix $\mathbf{W}^{\text{rate}}$ and exponentiating the resulting quantity. These factors are determined by multiplying the vector of activities $\mathbf{g}_t$ of the generator by a matrix $\mathbf{W}^{\text{fac}}$. The activities of the generator's units depend on two elements: a trial-specific initial state vector $\mathbf{g}_0$ (one for each trial), and the parameters defining the connections of the network (fixed across trials after training). The units of the generator do not correspond directly to any recorded channels, but, rather, the generator models the dynamics underlying the observed data. A linear readout of the activity of the encoder provides the inferred initial state $\mathbf{g}_0$. To compute $\mathbf{g}_0$ 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 backward and forward to compute $\mathbf{g}_0$, 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 Supplementary Fig. 1 we apply LFADS to a one-dimensional (1D) pendulum to show how LFADS operates for a simple, non-biological dynamical system). Thus, LFADS turns time series of single-trial recorded spike counts into low-dimensional dynamic factors and underlying rates that generate the observed spikes.

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 Supplementary Note 1; Lorenz attractor, Supplementary Fig. 2; chaotic RNN, Supplementary Fig. 3; chaotic RNN with pulse inputs, Supplementary Figs 7 and 8; and an RNN trained to integrate white noise, Supplementary Fig. 9). When compared on simulated data, LFADS outperforms a number of state-of-the-art machine-learning techniques (Gaussian process factor analysis (GPFA)[12], Poisson feed-forward neural network linear dynamical system (PfLDS)[15], and variational latent Gaussian process (vLGP)[13]; Supplementary Figs 2 and 3).

We then trained LFADS on multielectrode array data (single-trial spiking activity) from M1 and PMd, recorded while a monkey made reaching movements (see Methods for model training details, and Supplementary Table 1 for all model hyperparameters). The analyzed trials were 800-ms long and aligned to movement onset (that is, the time when arm movement was first detectable). We show inferred rates and factors for seven example trials (Fig. 1b).

We next experimentally validated LFADS-inferred factors and rates by reproducing features seen in common neuroscientific analyses (peri-stimulus time histograms (PSTHs), cross-correlations; Fig. 2, Supplementary Data 1–3); predicting held-out, simultaneously recorded neurons (Fig. 2); predicting details of behavior (Figs 2, 4, and 5 and Supplementary Fig. 6); showing single-trial features previously demonstrated in trial-averaged analysis (Fig. 3); predicting held-out conditions (Fig. 3); and correlating with LFPs (Fig. 6). In all examples we show, we trained LFADS to model observed spiking data from individual trials without any information about task conditions or behavioral parameters (for example, reach kinematics or electromyography (EMG)).

Validation of LFADS inferences using a complex reaching task. We applied LFADS to 202 neurons simultaneously recorded from M1 and PMd during a maze task (see Methods) in which a monkey made a variety of straight and curved reaches (Fig. 2a; the dataset consisted of ~2,300 individual reach trials spanning 108 reach types).

We first compared LFADS-inferred rates to smoothed spikes and to GPFA[12]-inferred rates (Fig. 2b). Condition-averaged smoothed spikes are commonly known as the PSTH; these assume that rates

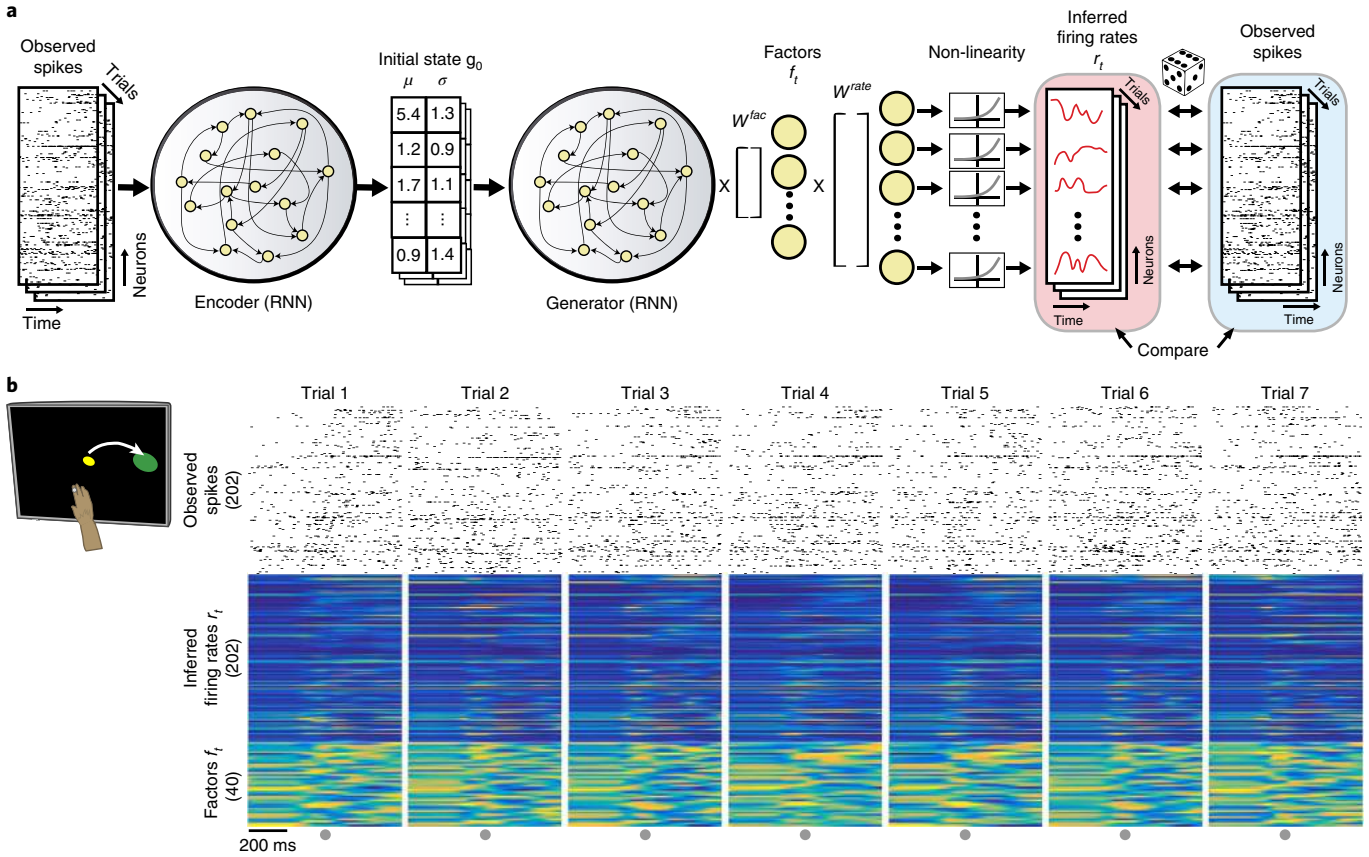

Fig. 1 | LFADS is a generative model that assumes that observed single-trial spiking activity is generated by an underlying dynamical system. a, Schematic overview of the LFADS architecture. Details are provided in the main text. **b**, Example spiking activity recorded from M1 and PMd as a monkey performed a reaching task, as well as the corresponding rates ***rt*** and factors ***ft*** inferred by LFADS (seven representative trials are shown). Circles denote time of movement onset.

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. LFADS assumes that rates are predictable—that is, 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. 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 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 (four neurons shown; all PSTHs are included as Supplementary Data 1). Finally, LFADS-inferred rates reproduce patterns of correlations across time (Supplementary Data 2) and neurons (Supplementary 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 fast temporal features better than GPFA.

LFADS encodes each individual trial by an initial state vector ($\mathbf{g}_0$). To test whether there was behaviorally relevant structure in the $\mathbf{g}_0$ encoding, we applied the non-linear dimensionality reduction technique t-distributed stochastic neighbor embedding (t-SNE; Fig. 2c) to reduce the dimensionality to 3 and color coded the ~2,300 points based on the angle of the target of the upcoming reach. t-SNE uncovered clear structure in the learned $\mathbf{g}_0$ encoding. Specifically, trials with similar kinematic structure are encoded with similar initial conditions (Supplementary Video 1). This demonstrates that the generator does not learn arbitrary sequences to model each trial, but instead learns 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). We estimated hand velocities from LFADS-inferred rates using cross-validated optimal linear estimation (OLE[26]). Using the full population of 202 neurons, decoding using LFADS-inferred rates outperformed results obtained by binning or smoothing spike trains, or by using GPFA (average R^2 of 0.90 across the dataset, versus 0.66, 0.69, and 0.34 for smoothing, GPFA, and binning, respectively; detailed in the Methods). We also determined performance as a function of population size by drawing random subsamples 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. We held the bin size and number of factors used by LFADS (5 ms and 20, respectively) constant for all models across all population sizes, while we chose the bin size and number of factors used for GPFA 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 (that is, 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

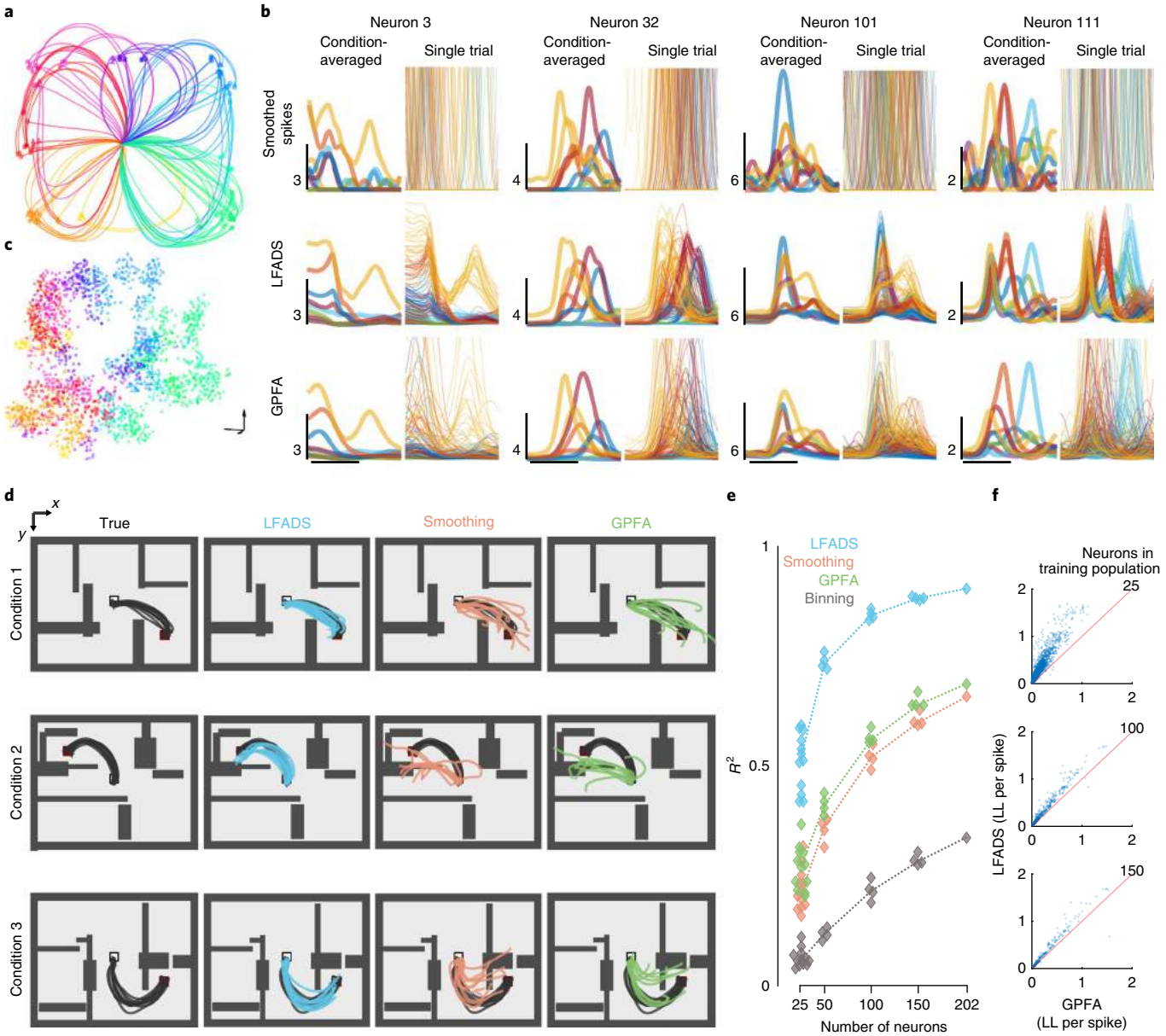

Fig. 2 | Application of LFADS to a maze reaching task. a, Individual reaches of a monkey during a maze reaching task, colored by target location. **b**, Comparison of condition-averaged (left) and single-trial (right) rates for four individual neurons (columns) for three different methods (rows). Left: 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 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 s^{-1}). PSTHs for all neurons are shown in Supplementary Data 1. **c**, Application of t-SNE to the generator initial conditions ($\mathbf{g}_0$). Each point represents the reduction of the $\mathbf{g}_0$ vector into a 3D t-SNE space for an individual trial (2,296 trials total); 2D projection shown, full 3D projection shown in Supplementary Video 1. Trials are color-coded as in **a**. **d**, Decoding reaching kinematics using OLE. Each row shows an example condition (3 shown of 108 total). Column 1: true reach trajectories (black traces, ten 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 via LFADS, by smoothing with a Gaussian filter (40-ms s.d.), or by 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 (2,296 trials), for all 3 techniques and additionally for simple binning of the neural data. Accuracy was also measured for random subsamples from the full neural population of 202 neurons. Dotted lines connect the median R^2 values for each population size. **f**, Performance of LFADS and GPFA in predicting responses of neurons held out from model training. Each point represents a given held-out neuron for a given random sampling of the population (same populations as in **e**). Performance was evaluated using log likelihood (LL) per spike[41].

counts in a cross-validated manner. For each held-out neuron, we trained a GLM to relate inferred factors to observed spike counts for a training subset of trials, and predicted rates for that neuron for a test subset using the trained GLM. The rates produced by LFADS-inferred factors predicted 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).

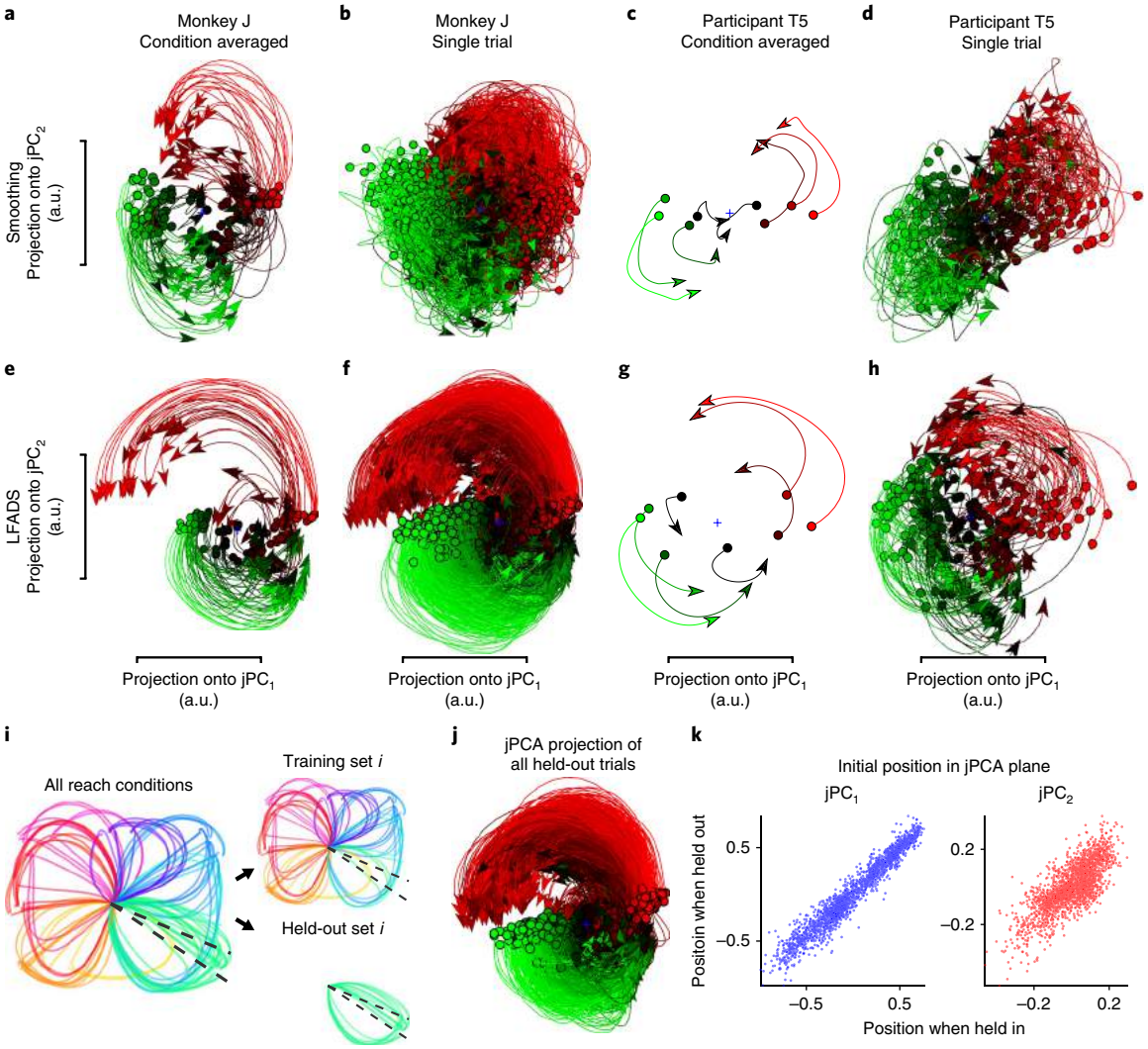

Fig. 3 | LFADS uncovers known rotational dynamics in monkey and human motor cortical activity on a single-trial basis. a,**c**, Condition-averaged neural population state trajectories in the M1 of monkeys and humans for a single task condition obtained with jPCA. a.u., arbitrary units; jPC_1 and jPC_2 are the first two compontents of jPCA (see main text). **b**,**d**, Same representation as in **a** and **c**, but for single-trial neural population activity. **e**,**g**, Condition-averaged inferred rates obtained with LFADS. **f**,**h**, Same representation as in **e** and **g** but for individual trials (monkey, 2,296 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. Then, 19 LFADS models' generator dynamics were 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 (that is, 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 (2,296 trials).

Uncovering rotational dynamics in M1. We next tested whether the population dynamics inferred by LFADS on single trials exhibited dynamic features that have previously been identified in trial-averaged data; specifically, the rotational dynamics underlying M1 and 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). We obtained these results by averaging the rate of each neuron across all trials corresponding to a particular reach condition and then applying a form of dimensionality reduction (j principal components analysis (jPCA)[3]). Although condition averaging reveals the basic oscillatory dynamics, single trials provide noisy and unstructured views of the neural trajectories (Fig. 3b,d). 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 (Fig. 3e,g), but also demonstrates, for the first time, the presence of rotational dynamics on single trials (Fig. 3f and Supplementary Video 2, monkey J, 2,296 maze reaching trials, and Fig. 3h, participant T5, 114 center-out movement attempts).

We then asked 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, we split data into training conditions and held-out (validation) conditions based on target angle (Fig. 3i). Briefly, we uniformly divided the workspace into

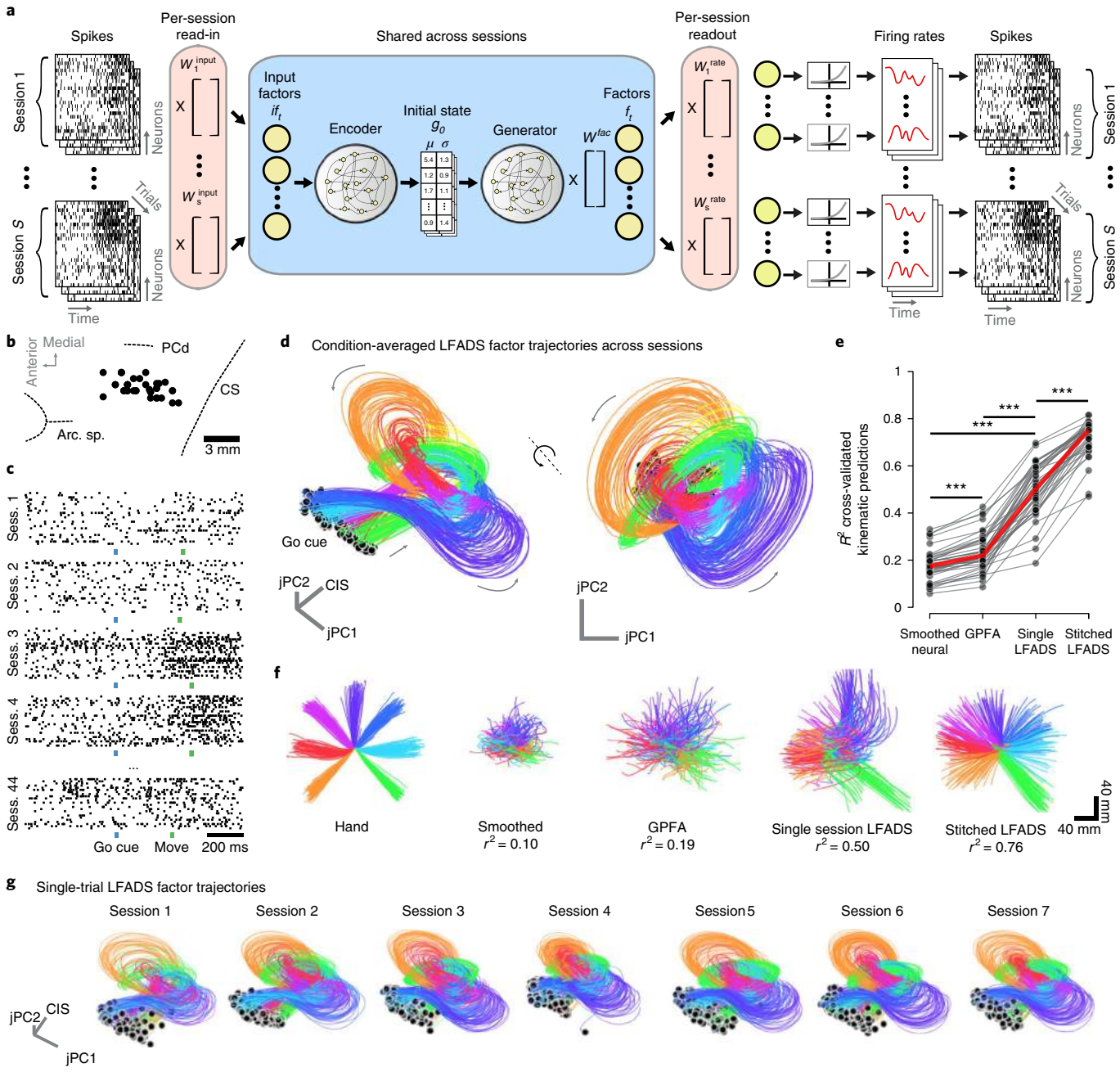

Fig. 4 | Using 'dynamic neural stitching', LFADS combines data from separately collected, non-overlapping recordings of the neural population by learning one consistent dynamical model. a, Schematic of the LFADS architecture adapted for dynamic neural stitching. Details are provided in the main text. For this example, each $\mathbf{W}_s^{rate}$ was learned, whereas $\mathbf{W}_s^{input}$ was set using a principal components regression approach (see 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 upward 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. Sess., session. **d**, Factor trajectories after training for each behavioral condition across recording sessions, produced by a multi-session stitched LFADS model. Traces are condition-averaged factor trajectories projected into a subspace which spans the 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 (44 trajectories of each color). **e**, R^2 values between arm kinematics and 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. ***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 7 of 44 sessions are shown for ease of presentation (see also Supplementary Video 3).

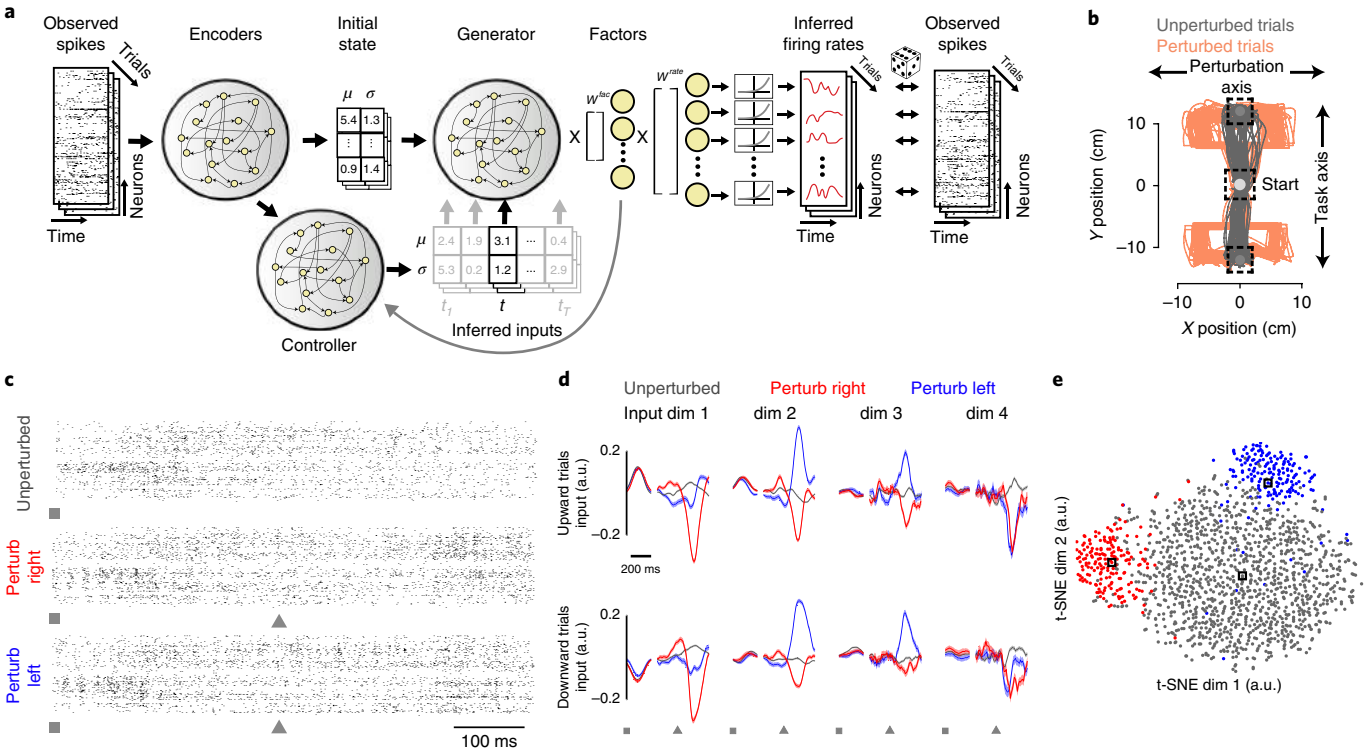

Fig. 5 | LFADS uncovers the presence, identity, and timing of unexpected perturbations in the cursor jump task. a, Schematic of the LFADS architecture adapted for inferring inputs to a neural population. LFADS reduces individual trials to an initial condition ($\mathbf{g}_0$) and a set of time-varying inferred inputs ($\mathbf{u}_t$), the latter of which are modeled stochastically with a mean and variance, which are inputted to the generator at each time point. The $\mathbf{u}_t$ is output by a controller RNN, which receives time-varying input from the encoding network, as well as the factor's 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 (gray 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 and 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 four inferred inputs to model the neural activity. For presentation, two trial alignments were used before 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; gray, unperturbed trials; blue, perturb left trials; red, perturb right trials). Around the time of target onset, the identity of the target (up versus down) is modeled by the inputs (for example, dimension 1). Around the time of the perturbation, LFADS used specific inferred input patterns to model each perturbation type (for example, dimensions 1 and 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). Dim, dimension. Black boxes denote locations in t-SNE space for the example trials shown in panel **c**.

angular bins and grouped conditions by the position of their reach target. This resulted in 19 sets of conditions (see Methods). For each set, we trained an LFADS model solely on the 18 other training condition sets, and then evaluated on the held-out set. We then collated LFADS-inferred rates for all of 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). 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, versus held-out, and found a clear correlation (Fig. 3k). This proof-of-principle analysis demonstrates that LFADS can learn dynamics that generalize to novel conditions, provided there are datasets with sufficient trial counts and diverse conditions to capture the neural population's dynamics.

Stitching together data from multiple sessions. Experiments are often performed across multiple sessions, with different neurons recorded on each session. LFADS can 'stitch' such data together to create a more powerful and comprehensive dynamical model. The aim is similar to previous efforts that relate separately recorded neural population activity[27,28], but, importantly, LFADS relates the separate sessions through a learned non-linear 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 can 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 $\mathbf{W}^{\text{input}}$, mapping from observed spiking to input factors, and 'read-out' matrices $\mathbf{W}^{\text{rate}}$, mapping from factors to neuron rates. The shapes of these matrices can vary to match the number of neural channels recorded in each dataset. Importantly, a single

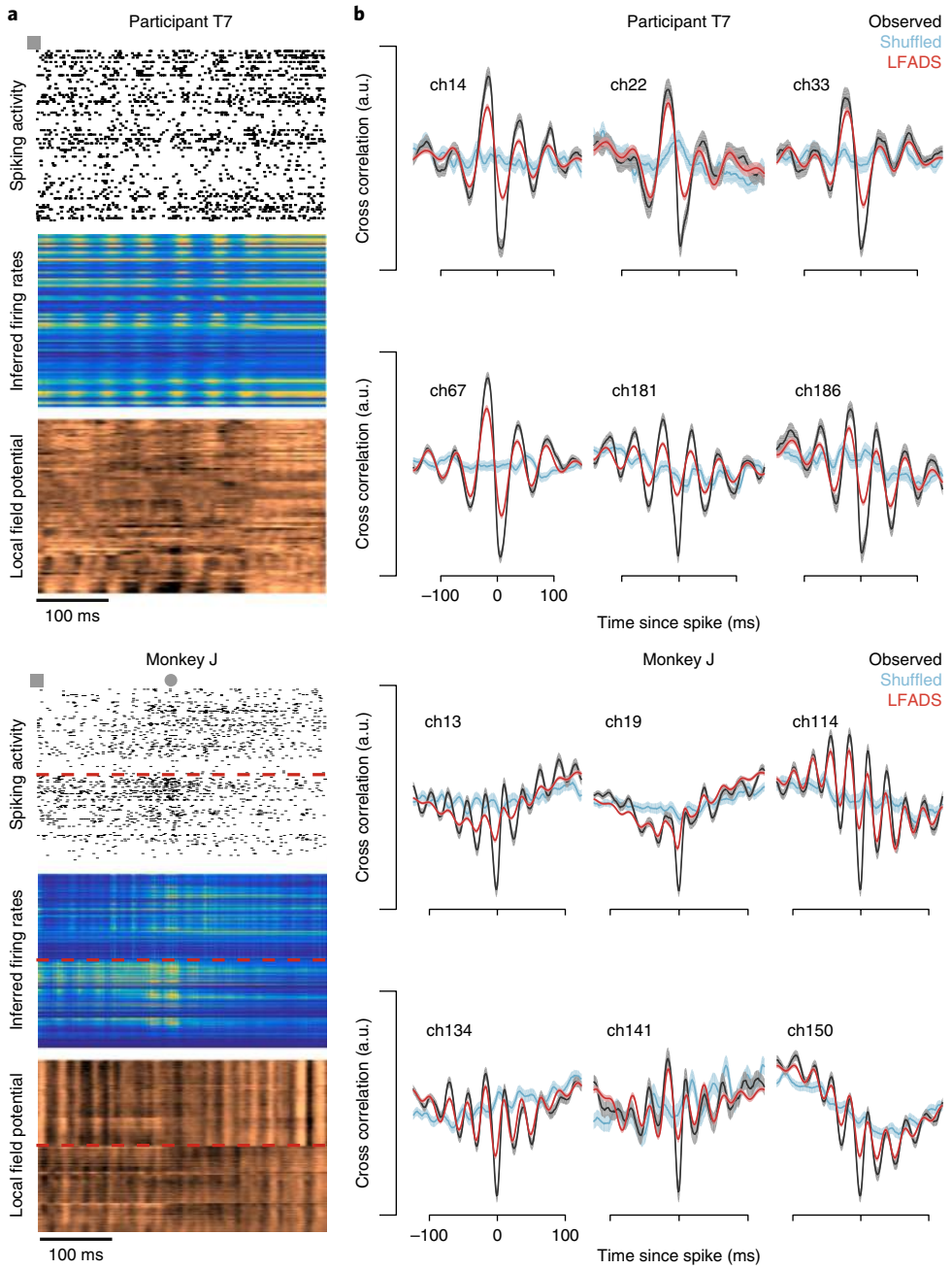

Fig. 6 | LFADS uncovers fast oscillatory structure in neural firing patterns. a, Example single-trial spiking activity recorded from human M1 and monkey M1 and PMd, as well as LFADS-inferred rates, and LFPs. Shown are 400 ms of data, beginning at the time of target presentation during an 8-target center-out-and-back movement paradigm. For T7, analyses were restricted to channels that showed significant modulation during movement attempts (78 of 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 LFPs 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). LFPs were first low-pass filtered (75-Hz cutoff frequency). Randomly shuffling the trial identity (that is, correlating spikes from one trial with LFP from another) largely removed the fast, oscillatory components in the cross-correlograms (blue traces). Ch, channel.

encoder, generator, and factor matrix $\mathbf{W}^{\text{fac}}$ 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 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 linear probes). We trained 1 stitched multi-session LFADS model on a combined dataset consisting of 44 recording sessions that spanned 162 d (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, Supplementary Fig. 5, and Supplementary 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 ($\mathbf{f}_t$) 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, probably arises from the difference in recording methodologies. We also predicted reaction time from LFADS factors (Supplementary 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); that is, we introduce inputs to allow the neural population activity to be modeled as a non-autonomous dynamical system (Figs 5–6 and Supplementary Figs 7–9). 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 that supplies the deviation from the unperturbed dynamics (we outline caveats in the Discussion). This means that, beyond Poisson spiking, trial-to-trial variability is captured by both the initial condition $\mathbf{g}_0$ and the inferred input $\mathbf{u}_t$ 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, toward upward or downward targets (monkey J; see Methods). 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 and PMd (Fig. 5c), allowing four inferred inputs (choice of dimensionality detailed in the 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. Before 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; for example, Input dimension 1, comparing inputs inferred for upward versus downward trials), but are not distinct with respect to perturbation type (that is, red, blue, and gray traces are overlapping), as perturbations occurred 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 gray traces; for example, Input dimension 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 was similar across downward and upward reaches (Fig. 5d, top and bottom panels). The trends were also visible on single trials (Supplementary 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 that LFADS can predict, on average, the presence, identity, and timing of inputs to M1 related to task perturbations.

LFADS rate oscillations correlate with LFPs. Another known dynamic feature of motor cortical activity is the rhythmic spiking that often occurs during the premovement period, typically phase-locked to accompanying LFP oscillations (15–40 Hz; for example, see refs. [29,30]). 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[31]. 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 we did not give the model 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). We computed cross-correlations 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 LFPs on single trials, which was 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 LFPs 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 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 ref. [32]). LFADS builds

empirical models of the non-linear 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, strongly suggests 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. 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 distinguishes the dynamics internal to a neural circuit from the influence of unmeasured input from other brain regions, which is a challenge in neuroscience. Although the nature of the inputs inferred by LFADS informs 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 (for example, biophysical spiking versus 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, 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 when inputs are inferred, as there is potential for the system to forego modeling dynamics altogether when reconstructing the data. Specifically, 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 actually modeling any of the underlying dynamics. Due to this confound, reconstruction quality is not an ideal metric for evaluating the model's inference of underlying population 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 will be helpful in understanding the role of computation through dynamics in 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 BMIs, which decode movement intention in real-time to control external devices[33–36], or closed-loop neuromodulation approaches, which require real-time neural state estimates to guide stimulation[37–40]. In addition, the ability of stitching to improve neural state estimates by combining multiple recordings may improve stability of these devices. Taken together, LFADS has the potential to help understand neural computation and dynamics, and to apply this knowledge towards the treatment of diverse neurological disorders.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, statements of data availability and associated accession codes are available at https://doi.org/10.1038/s41592-018-0109-9.

Received: 28 February 2018; Accepted: 28 June 2018;
Published online: 17 September 2018

References

1. Afshar, A. et al. Single-trial neural correlates of arm movement preparation. *Neuron* **71**, 555–564 (2011).
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).
3. Churchland, M. M. et al. Neural population dynamics during reaching. *Nature* **487**, 51–56 (2012).
4. Harvey, C. D., Coen, P. & Tank, D. W. Choice-specific sequences in parietal cortex during a virtual-navigation decision task. *Nature* **484**, 62–68 (2012).
5. Kaufman, M. T., Churchland, M. M., Ryu, S. I. & Shenoy, K. V. Cortical activity in the null space: permitting preparation without movement. *Nat. Neurosci.* **17**, 440–448 (2014).
6. Kobak, D. et al. Demixed principal component analysis of neural population data. *Elife* **5**, e10989 (2016).
7. Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013).
8. Pandarinath, C. et al. Neural population dynamics in human motor cortex during movements in people with ALS. *Elife* **4**, e07436 (2015).
9. Sadtler, P. T. et al. Neural constraints on learning. *Nature* **512**, 423–426 (2014).
10. Shenoy, K. V., Sahani, M. & Churchland, M. M. Cortical control of arm movements: a dynamical systems perspective. *Annu. Rev. Neurosci.* **36**, 337–359 (2013).
11. Ahrens, M. B. et al. Brain-wide neuronal dynamics during motor adaptation in zebrafish. *Nature* **485**, 471–477 (2012).
12. Yu, B. M. et al. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *J. Neurophysiol.* **102**, 614–635 (2009).
13. Zhao, Y. & Park, I. M. Variational latent Gaussian process for recovering single-trial dynamics from population spike trains. *Neural Comput.* **29**, 1293–1316 (2017).
14. Aghagolzadeh, M. & Truccolo, W. Latent state-space models for neural decoding. *Conf. Proc. IEEE Eng.Med. Biol. Soc.* **2014**, 3033–3036 (2014).
15. Gao, Y., Archer, E. W., Paninski, L. & Cunningham, J. P. Linear dynamical neural population models through nonlinear embeddings. In *Proc. 30th International Conference on Neural Information Processing Systems* (eds. Lee, D. D. et al.) 163–171 (Curran Associates, Red Hook, NY, 2016).
16. Kao, J. C. et al. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. *Nat. Commun.* **6**, 7759 (2015).
17. Macke, J. H. et al. Empirical models of spiking in neural populations. In *Advances in Neural Information Processing Systems 24* (eds Shawe-Taylor, J. et al.) 1350–1358 (Curran Associates, Red Hook, NY, 2011).
18. Linderman, S. et al. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Proc. 20th International Conference on Artificial Intelligence and Statistics* vol. 54 (eds Singh, A. & Zhu, J.) 914–922 (PMLR/Microtome Publishing, Brookline, MA, 2017).
19. Petreska, B. et al. Dynamical segmentation of single trials from population neural data. In *Advances in Neural Information Processing Systems 24* (eds Shawe-Taylor, J. et al.) 756–764 (Curran Associates, Red Hook, NY, 2011).
20. Kato, S. et al. Global brain dynamics embed the motor command sequence of Caenorhabditis elegans. *Cell* **163**, 656–669 (2015).
21. Kaufman, M. T. 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).
23. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *arXiv* Preprint at https://arxiv.org/abs/1312.6114 (2013).
24. Doersch, C. Tutorial on variational autoencoders. *arXiv* Preprint at https://arxiv.org/abs/1606.05908 (2016).
25. Sussillo, D., Jozefowicz, R., Abbott, L. F. & Pandarinath, C. LFADS—latent factor analysis via dynamical systems. *arXiv* Preprint at https://arxiv.org/abs/1608.06315 (2016).
26. Salinas, E. & Abbott, L. F. Vector reconstruction from firing rates. *J. Comput. Neurosci.* **1**, 89–107 (1994).
27. Turaga, S. et al. Inferring neural population dynamics from multiple partial recordings of the same neural circuit. In *Advances in Neural Information Processing Systems 26* (eds Burges, C. J. C. et al.) 539–547 (Curran Associates, Red Hook, NY, 2013).
28. Nonnenmacher, M., Turaga, S. C. & Macke, J. H. Extracting low-dimensional dynamics from multiple large-scale neural population recordings by learning to predict correlations. In *Advances in Neural Information Processing Systems 30* (eds Guyon, I. et al.) 5702–5712 (Curran Associates, Red Hook, NY, 2017).
29. Donoghue, J. P., Sanes, J. N., Hatsopoulos, N. G. & Gaal, G. Neural discharge and local field potential oscillations in primate motor cortex during voluntary movements. *J. Neurophysiol.* **79**, 159–173 (1998).
30. Murthy, V. N. & Fetz, E. E. Synchronization of neurons during local field potential oscillations in sensorimotor cortex of awake monkeys. *J. Neurophysiol.* **76**, 3968–3982 (1996).
31. Fries, P. A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. *Trends. Cogn. Sci.* **9**, 474–480 (2005).
32. Yuste, R. From the neuron doctrine to neural networks. *Nat. Rev. Neurosci.* **16**, 487 (2015).
33. Gilja, V. et al. Clinical translation of a high-performance neural prosthesis. *Nat. Med.* **21**, 1142 (2015).
34. Pandarinath, C. et al. High performance communication by people with paralysis using an intracortical brain-computer interface. *Elife* **6**, e18554 (2017).
35. Sussillo, D. et al. A recurrent neural network for closed-loop intracortical brain–machine interface decoders. *J. Neural. Eng.* **9**, 26027 (2012).
36. Sussillo, D., Stavisky, S. D., Kao, J. C., Ryu, S. I. & Shenoy, K. V. Making brain–machine interfaces robust to future neural variability. *Nat. Commun.* **7**, 13749 (2016).
37. Ezzyat, Y. et al. Closed-loop stimulation of temporal cortex rescues functional networks and improves memory. *Nat. Commun.* **9**, 365 (2018).
38. Klinger, N. V. & Mittal, S. Clinical efficacy of deep brain stimulation for the treatment of medically refractory epilepsy. *Clin. Neurol. Neurosurg.* **140**, 11–25 (2016).
39. Little, S. et al. Adaptive deep brain stimulation in advanced Parkinson disease. *Ann. Neurol.* **74**, 449–457 (2013).
40. Rosin, B. et al. Closed-loop deep brain stimulation is superior in ameliorating parkinsonism. *Neuron* **72**, 370–384 (2011).
41. Williamson, R. S., Sahani, M. & Pillow, J. W. The equivalence of information-theoretic and likelihood-based methods for neural dimensionality reduction. *PLoS. Comput. Biol.* **11**, e1004141 (2015).

Acknowledgements

We thank J.P. Cunningham and J. Sohl-Dickstein for extensive conversation. We thank M.M. Churchland for contributions to data collection for monkey J, C. Blabe and P. Nuyujukian for assistance with research sessions with participant T5, E. Eskandar for array implantation with participant T7, and B. Sorice and A. Sarma for assistance with research sessions with participant T7. 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 an NIH-NIDCD award (R01DC014034). K.V.S. and J.M.H.'s research was supported by Stanford BioX-NeuroVentures, Stanford Institute for Neuro-Innovation and Translational Neuroscience, the Garlick Foundation, and the Reeve Foundation. L.R.H.'s research was supported by an NIH-NIDCD award (R01DC009899), the Rehabilitation Research and Development Service, Department of Veterans Affairs (B6453R), the MGH-Deane Institute for Integrated Research on Atrial Fibrillation and Stroke, and the 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.

Author contributions

C.P., D.J.O., and D.S. designed the study, performed analyses, and wrote the manuscript with input from all other 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 revision of the manuscript.

Competing 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. D.S. and J.C. are employed by Google Inc., and R.J. was employed by Google Inc. at the time the research was conducted. This funder provided support in the form of salaries for authors, but did not have any additional role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41592-018-0109-9.

Correspondence and requests for materials should be addressed to C.P. or D.S.

Methods

The LFADS model. *The VAE.* The LFADS model is an instantiation of a VAE[23,42] extended to sequences, as in ref. [43] or ref. [44]. 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 $Q(\mathbf{z}|\mathbf{x})$. If data generation is desired, a data sample $\mathbf{x}$ may then be 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 non-linear 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 $P(\mathbf{z})$; that is, 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 ref. [43], 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 an RNN receiving an input as $\text{state}_t = \text{RNN}^a(\text{state}_{t-1}, \text{input}_t)$, for an RNN named '$a$'. It is understood that if there are two network modules, such as RNNs, with different names (for example, $\text{RNN}^a(.,.)$ and $\text{RNN}^b(.,.)$), these network modules do not share parameters.

The LFADS generative model. 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 $\mathbf{a}_{1:T}$ are not themselves being modeled, but 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 dataset, $\{\mathbf{x}_{1:T}, \mathbf{a}_{1:T}\}$, and the part of the dataset 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 non-linearity, $\mathbf{r}_{1:T} = \exp(\mathbf{W}^{\text{rate}}(\mathbf{f}_{1:T}))$. 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; for example, refs. [3,7,20], and see ref. [22] for a more complete discussion.

The factors are generated by a recurrent non-linear neural network and are characterized by an affine transformation of its state vector, $\mathbf{f}_{1:T} = \mathbf{W}^{\text{fac}}(\mathbf{g}_{1:T})$, with $\mathbf{g}_t$ of dimension N. Running the network requires an initial condition $\mathbf{g}_0$, which is drawn from a prior distribution $P^{g_0}(\mathbf{g}_0)$. 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, 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. 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 $P^{u_1}(\mathbf{u}_1)$ defining the distribution over $\mathbf{u}_1$ (see Methods, "Autoregressive prior for inferred input").

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 $\mathbf{g}_0$

$$\hat{g}_0 \sim P^{g_0}(\mathbf{g}_0) = \mathcal{N}(0, \kappa\mathbf{I}) \tag{3}$$

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,

$$\hat{\mathbf{u}}_t \sim \begin{cases} P^{u_1}(\mathbf{u}_1) & \text{if } t = 1 \\ P^{u}(\mathbf{u}_t|\mathbf{u}_{t-1}) & \text{otherwise} \end{cases} \tag{4}$$

$$\mathbf{g}_t = \text{RNN}^{\text{gen}}(\mathbf{g}_{t-1}, \hat{\mathbf{u}}_t) \tag{5}$$

$$\mathbf{f}_t = \mathbf{W}^{\text{fac}}(\mathbf{g}_t) \tag{6}$$

$$\mathbf{r}_t = \exp(\mathbf{W}^{\text{rate}}(\mathbf{f}_t)) \tag{7}$$

$$\hat{\mathbf{x}}_t \sim \text{Poisson}(\mathbf{x}_t|\mathbf{r}_t) \tag{8}$$

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 priors 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 Methods, "Autoregressive prior for inferred input," for more details). We chose the gated recurrent unit (GRU)[45] as our recurrent function for all of the networks we use (see section "GRU equations" in the Methods 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 (5). Note that doing so will make the generation process necessarily dependent on including an observed input. The generator model is illustrated in Supplementary 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.

The 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 Supplementary Fig. 12 and below). We begin by describing the network that defines $Q^{g_0}(\mathbf{g}_0|\mathbf{x},\mathbf{a})$. Its mean and variance are given in terms of a vector $\mathbf{E}^{\text{gen}}$ by

$$\mu^{g_0} = \mathbf{W}^{\mu^{g_0}}(\mathbf{E}^{\text{gen}}) \tag{9}$$

$$\sigma^{g_0} = \exp\left(\frac{1}{2}\mathbf{W}^{\sigma^{g_0}}(\mathbf{E}^{\text{gen}})\right) \tag{10}$$

$\mathbf{E}^{\text{gen}}$ 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 backward (from $t = T$ to $t = 1$),

$$\mathbf{e}_t^{\text{gen},b} = \text{RNN}^{\text{gen},b}(\mathbf{e}_{t+1}^{\text{gen},b}, [\mathbf{x}_t, \mathbf{a}_t]) \tag{11}$$

$$\mathbf{e}_t^{\text{gen},f} = \text{RNN}^{\text{gen},f}(\mathbf{e}_{t-1}^{\text{gen},f}, [\mathbf{x}_t, \mathbf{a}_t]) \tag{12}$$

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

$$\mathbf{E}^{\text{gen}} = [\mathbf{e}_1^{\text{gen},b}, \mathbf{e}_T^{\text{gen},f}] \tag{13}$$

Running the encoding network both forward and backward in time allows $\mathbf{E}^{\text{gen}}$ 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:

$$\hat{\mathbf{g}}_0 \sim Q^{g_0}(\mathbf{g}_0|\mathbf{x},\mathbf{a}) = \mathcal{N}(\mathbf{g}_0 \mid \boldsymbol{\mu}^{g_0}, \boldsymbol{\sigma}^{g_0}) \tag{14}$$

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 $\mathbf{e}_t^{\mathrm{con},b}$ and $\mathbf{e}_t^{\mathrm{con},f}$ that are defined by equations identical to equations (11) and (12) (although with different trainable network parameters). Finally, the time-dependent input to the controller RNN is defined as

$$\mathbf{E}_t^{\mathrm{con}} = [\mathbf{e}_t^{\mathrm{con},b}, \mathbf{e}_t^{\mathrm{con},f}] \tag{15}$$

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,

$$\mathbf{c}_t = \mathrm{RNN}^{\mathrm{con}}(\mathbf{c}_{t-1}, [\mathbf{E}_t^{\mathrm{con}}, \mathbf{f}_{t-1}]) \tag{16}$$

Thus, the controller is privy to the information about $\mathbf{x}_{1:T}$ and $\mathbf{a}_{1:T}$ encoded in the variable $\mathbf{E}_t^{\mathrm{con}}$, 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, $\boldsymbol{c}_0$, 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, $\boldsymbol{c}_t$,

$$\hat{\mathbf{u}}_t \sim Q^u(\mathbf{u}_t|\mathbf{x},\mathbf{a}) = \mathcal{N}(\mathbf{u}_t|\boldsymbol{\mu}_t^u, \boldsymbol{\sigma}_t^u) \tag{17}$$

with

$$\boldsymbol{\mu}_t^u = \mathbf{W}^{\mu^u}(\mathbf{c}_t) \tag{18}$$

$$\sigma_t^u = \exp\left(\frac{1}{2}\mathbf{W}^{\sigma^u}(\mathbf{c}_t)\right) \tag{19}$$

We control the information flow out of the controller and into the generator by applying a regularizer on $\mathbf{u}_t$ (a Kullback–Leibler divergence term, described in this Methods section under "The loss function" and "Hyper-parameters and further details of LFADS implementation"), and also by explicitly limiting the dimensionality of $\mathbf{u}_t$, the latter of which is controlled by a hyperparameter.

The full LFADS inference model. The full LFADS model (Supplementary 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, $\hat{\mathbf{g}}_0 \sim \mathcal{N}(\mathbf{g}_0 \mid \boldsymbol{\mu}^{g_0}, \boldsymbol{\sigma}^{g_0})$. Then, for each time step from 1 to T, the generator is updated, as well as the factors and rates, according to

$$\mathbf{c}_t = \mathrm{RNN}^{\mathrm{con}}(\mathbf{c}_{t-1}, [\mathbf{E}_t^{\mathrm{con}}, \mathbf{f}_{t-1}]) \tag{20}$$

$$\boldsymbol{\mu}_t^u = \mathbf{W}^{\mu^u}(\mathbf{c}_t) \tag{21}$$

$$\boldsymbol{\sigma}_t^u = \exp\left(\frac{1}{2}\mathbf{W}^{\sigma^u}(\mathbf{c}_t)\right) \tag{22}$$

$$\hat{\mathbf{u}}_t \sim \mathcal{N}(\mathbf{u}_t \mid \boldsymbol{\mu}_t^u, \boldsymbol{\sigma}_t^u) \tag{23}$$

$$\mathbf{g}_t = \mathrm{RNN}^{\mathrm{gen}}(\mathbf{g}_{t-1}, \hat{\mathbf{u}}_t) \tag{24}$$

$$\mathbf{f}_t = \mathbf{W}^{\mathrm{fac}}(\mathbf{g}_t) \tag{25}$$

$$\mathbf{r}_t = \exp(\mathbf{W}^{\mathrm{rate}}(\mathbf{f}_t)) \tag{26}$$

$$\hat{\mathbf{x}}_t \sim \mathrm{Poisson}(\mathbf{x}_t|\mathbf{r}_t) \tag{27}$$

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 (27) is not executed, unless one explicitly desires to generate spikes.

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,

$$\log P(\mathbf{x}_{1:T}) \geq \mathcal{L} = \mathcal{L}^x - \mathcal{L}^{KL} \tag{28}$$

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

$$\mathcal{L}^x = \left\langle \sum_{t=1}^{T} \log(\mathrm{Poisson}(\mathbf{x}_t|\mathbf{r}_t)) \right\rangle_{\mathbf{g}_0,\mathbf{u}_{1:T}} \tag{29}$$

$$\begin{aligned} \mathcal{L}^{\mathrm{KL}} = & \left\langle D_{\mathrm{KL}}(\mathcal{N}(\mathbf{g}_0|\boldsymbol{\mu}^{g_0}, \boldsymbol{\sigma}^{g_0}) \| P^{g_0}(\mathbf{g}_0)) \right\rangle_{\mathbf{g}_0} + \\ & \left\langle D_{\mathrm{KL}}(\mathcal{N}(\mathbf{u}_1|\boldsymbol{\mu}_1^u, \boldsymbol{\sigma}_1^u) \| P^{u_1}(\mathbf{u}_1)) \right\rangle_{\mathbf{g}_0,\mathbf{u}_1} + \\ & \left\langle \sum_{t=2}^{T} D_{\mathrm{KL}}(\mathcal{N}(\mathbf{u}_t|\boldsymbol{\mu}_t^u, \boldsymbol{\sigma}_t^u) \| P^u(\mathbf{u}_t|\mathbf{u}_{t-1})) \right\rangle_{\mathbf{g}_0,\mathbf{u}_{1:T}} \end{aligned} \tag{30}$$

where the brackets denote marginalizations over the subscripted variables. Evaluating the $T+1$ Kullback–Leibler 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 ref. [23]. We minimize the negative bound, $-\mathcal{L}$, using the reparameterization trick for Gaussian distributions to backpropagate low-variance, unbiased gradient estimates[23]. These gradients are used to train the system in an end-to-end fashion, as is typically done in deterministic settings.

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 $\boldsymbol{h}_t$, the hidden state, at time t, the GRU update equation, $\boldsymbol{h}_t = \mathrm{GRU}(\mathbf{x}_t, \boldsymbol{h}_{t-1})$, is defined as

$$\mathbf{r}_t = \sigma(\mathbf{W}^r([\mathbf{x}_t, \mathbf{h}_{t-1}])) \tag{31}$$

$$\mathbf{u}_t = \sigma(\mathbf{W}^u([\mathbf{x}_t, \mathbf{h}_{t-1}])) \tag{32}$$

$$\mathbf{c}_t = \tanh(\mathbf{W}^c([\mathbf{x}_t, \mathbf{r}_t \odot \mathbf{h}_{t-1}])) \tag{33}$$

$$\mathbf{h}_t = \mathbf{u}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \odot \mathbf{c}_t \tag{34}$$

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

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

$$s(t) = \alpha s(t-1) + \epsilon_s(t) \tag{35}$$

with $0 \leq \alpha < 1$ and noise variable $\epsilon_s(t)$ drawn from $\mathcal{N}(0, \sigma_\epsilon^2)$. An equivalent formulation for the 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 $\sigma_\epsilon^2 = \sigma_p^2(1-\alpha^2)$. To make the process distribution stationary the correct distribution for $s(0)$ is $\mathcal{N}(0, \sigma_p^2)$. 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 ith element of $\mathbf{u}_t$ an autocorrelation τ_i and process variance $\sigma_{p,i}^2$ are initialized to user-defined initial values.

Modifications to the LFADS algorithm for stitching together data from multiple recording sessions. To accommodate multiple recordings sessions, as in Fig. 4, 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 of 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 $\mathbf{W}^{\text{fac}}$ 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 (7), replacing it with equation (36) to change matrices as a function of recording session, thus introducing a session index, s, into the notation

$$\mathbf{r}_{s,t} = \exp(\mathbf{W}^{\text{rate}}_s(\mathbf{f}_{s,t})) \tag{36}$$

where the dimensions of $\mathbf{W}^{\text{rate}}_s$ 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 (11) and (12)), single trial by single trial. To handle multiple sessions' data, we modify this practice by introducing a per-session read-in matrix, $\mathbf{W}^{\text{input}}_s$. These read-in matrices map from recorded units to 'input factors'. Then, for the bidirectional encoding RNN for $\mathbf{g}_0$, we modified equations (11) and (12) by inputting the linearly transformed spikes, yielding

$$\mathbf{e}^{\text{gen},b}_{s,t} = \text{RNN}^{\text{gen},b}(\mathbf{e}^{\text{gen},b}_{s,t+1}, [\mathbf{W}^{\text{input}}_s(\mathbf{x}_{s,t}), \mathbf{a}_t]) \tag{37}$$

$$\mathbf{e}^{\text{gen},f}_{s,t} = \text{RNN}^{\text{gen},f}(\mathbf{e}^{\text{gen},f}_{s,t-1}, [\mathbf{W}^{\text{input}}_s(\mathbf{x}_{s,t}), \mathbf{a}_t]) \tag{38}$$

where the dimensions of $\mathbf{W}^{\text{input}}_s(\cdot)$ 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 dimensions equal to the total number of units by 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 an $F \times T$ matrix of principal component scores. For each session, we regressed the matrix of principal component scores against the condition-averaged firing rates recorded in that session. The resulting matrix of regression coefficients, which best reconstructs a set of shared principal component scores from each session's firing rates, was used as the initial read-in matrix for that session, $\mathbf{W}^{\text{input}}_s$. The read-out matrix $\mathbf{W}^{\text{rate}}_s$ for the session was initialized to the transpose of $\mathbf{W}^{\text{input}}_s$. 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; for example, 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 (for example, 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 backpropagated. As a result, all shared parameters (for example, the encoder and generator RNN parameters and factor read-out matrix $\mathbf{W}^{\text{fac}}$) 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 are used for training.

Hyper-parameters and further details of LFADS implementation. A table of the major hyper-parameters for each model is listed in Supplementary 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[46] to the inputs and to a few feed-forward (input) connections[47] in the LFADS model. Specifically, we used dropout 'layers' around equation (13), around the input in equation (20), and around equation (24) from $\mathbf{g}_t$ to $\mathbf{f}_t$.

We added an L_2 penalty to recurrent portions of the generator (equations (31–34)) and controller networks to encourage simple dynamics. Specifically, we regularized any matrix parameter by which $\boldsymbol{h}_{t-1}$ was multiplied, but not those that multiplied $\mathbf{x}_t$.

As defined in equation (30), there is an information-limiting regularizer placed on $\mathbf{u}_t$ by virtue of minimizing the Kullback–Leibler divergence between the approximate posterior over $\mathbf{u}_t$ and the uninformative auto-regressive prior.

Following ref. [48], we added a linearly increasing schedule on the Kullback–Leibler divergence penalty so that the optimization does not quickly (and pathologically) set the Kullback–Leibler divergence to 0. By 2,000 training steps, the schedule reached the maximum value of the Kullback–Leibler 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 non-linearities.

The auto-regressive prior parameters were optimized to reduce the Kullback–Leibler 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 six 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 1×10^{-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 $\boldsymbol{h}_t$ 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 $\mathbf{W}^{\text{fac}}(\cdot)$ 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.

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 (for example, the rates, $\mathbf{r}_t$). For example, in de-noising 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 (equation (14) and (17)) some number of times (for example, 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, for example, $\bar{\mathbf{r}}_t \equiv \langle \mathbf{r}_t \rangle_{\mathbf{g}_0, \mathbf{u}_{1:T}}$. It is posterior averages such as $\bar{\mathbf{r}}_t$ that are shown in the majority of figures.

Related work in the machine-learning literature. A discussion of related works is presented in Supplementary Note 2.

Datasets. *Synthetic datasets.* Details for the synthetic datasets are presented in Supplementary Note 1.

Research participants with paralysis. See Supplementary Table 2 for recording technologies for all recorded data.

Permission for these studies was granted by the US Food and Drug Administration (Investigational Device Exemption) and Institutional Review Boards of Stanford University (protocol no. 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[49]. Informed consent, including consent to publish, was obtained from the participants before 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 amyotrophic lateral sclerosis (ALS) and had resultant motor impairment (ALS Functional Rating Scale Revised score of 17). In July 2013, participant T7 had two 96-channel intracortical silicon microelectrode arrays (1.5-mm electrode length; Blackrock Microsystems) implanted in the hand area of dominant M1. T7 retained very limited and inconsistent finger movements. Data reported are from T7's postimplant 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 American Spinal Cord Injury Association (ASIA) C injury classification, which occurred approximately 9 years before 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 August of 2016, participant T5 had two 96-channel intracortical silicon micro-electrode arrays (1.5-mm electrode length; Blackrock Microsystems) implanted in the upper extremity area of dominant M1. Data reported are from T5's postimplant day 51.

Research participants—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[8,33,34]. 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). The cursor began in the center of the screen, and targets would appear in one of eight 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 0.3–0.6 cm 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 7,500 Hz and then processed to obtain multi-unit 'threshold crossings'; that is, 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 root mean square (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 modulation that correlated with movement direction 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 (MathWorks), enabling millisecond timing precision for all computations. Neural data collected by the NeuroPort System (Blackrock Microsystems) 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 a liquid-crystal display 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.

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

Non-human primates—maze task. An adult male macaque monkey (monkey J) was trained to sit head-fixed in a primate chair and perform two-dimensional (2D) target acquisition tasks in a fronto-parallel plane by controlling an on-screen cursor with his hand movements. Monkey J was implanted with 2 96-electrode arrays (1-mm electrodes spaced 400 μm apart; Blackrock Microsystems) using standard neurosurgical techniques. The arrays were implanted into M1 and dorsal 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 2D workspace while the positions of the right index and middle fingertips were 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 to 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 2,296 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 (Supplementary Data 2 and 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 before 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:

$$\Sigma_T^c = \sum_{n=1}^{N} X(:,n,c)X(:,n,c)^T \tag{39}$$

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

$$\Sigma_N^c = \sum_{t=1}^{T} X(t,:,c)X(t,:,c)^T \tag{40}$$

Neural cross-correlation matrices were sorted using a MATLAB implementation of the Bron–Kerbosch maximal clique 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 95th 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.

Non-human primates—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). 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 ref. [50]. 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). 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 s to receive a liquid reward and success tone.

Voltage signals from each of the electrodes were band-pass filtered from 250 to 7,500 Hz and then processed to obtain multi-unit threshold crossings; that is, 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 cursor jump (Fig. 5) and LFP analyses (Fig. 6), 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 at 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 toward radial targets. On these randomly selected perturbation trials, during the monkey's reaching movement, the cursor position jumped; that is, it was offset by 6 cm perpendicular to the vertical axis. The jump happened after the cursor traveled 6 cm toward 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 5,140 trials. For each trial, 800 ms of data were taken, with data aligned to the start of the trial (target onset). In cases of perturbations, most jumps happened between 400 and 550 ms post target onset. The model was allowed to infer four inputs to the generator in order to fit the data. The choice of four inputs reflects three key facts about the system and task. First, we know that high-frequency oscillatory dynamics are present in the firing rates (for example, 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: before target onset, the subject does not know whether an upward or downward target will appear. Thus, the arrival of target position information to M1 is a 1D perturbation (upward or downward) 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 1D perturbation for the system to model. Thus, we reasoned that four inputs were a reasonable choice for modeling this particular recording configuration and task.

Non-human primates—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 and xylazine anesthesia, targeting an area in PMd that 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 three-dimensional (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 four 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 end point 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 that 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 a liquid-crystal display 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 toward 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 eight 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, second-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 probably spanned the layers of the cortex based on properties of the neural signals. We allowed 45–90 min 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 × r.m.s. 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 M1 and dorsal PMd. A 1,200-ms time window beginning 500 ms before the go cue to 700 ms afterward was chosen from each successful trial and used to train the LFADS model.

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 analysis methods are standard, but we provide references and operating parameters here.

Figure 2—application of LFADS to a maze reaching task. For the PSTHs and single-trial inferred and estimated firing rates in Fig. 2b, each trial was aligned by movement onset (that is, the time at which movement of the arm is first detectable), and the data analyzed began 400 ms before movement onset and ended 400 ms after movement onset. Data were preprocessed 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 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 and Supplementary Video 1), the initial conditions vector inferred by LFADS ($\mathbf{g}_0$) for each trial was mapped onto a low-dimensional subspace using t-SNE[51]. Three 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 3D t-SNE space, with colors corresponding to the end point of the reaching movement (Fig. 2a), and marker type corresponding to the type of reach (that is, markers used were circles, squares, and triangles, for straight, curved counter-clockwise, and curved clockwise reaches, respectively).

To decode kinematics from neural features (Fig. 2d,e), we used OLE[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 before 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 were optimized to maximize decoding accuracy, with binsize swept from 5 to 20 ms, and latents swept from 5 to 40 factors (see Supplementary 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 fivefold 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 fivefold 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 BMI decoding approaches such as the Kalman filter. This relationship is outlined in ref. [52]; 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 (that is, 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 (for example, the feedback intention-trained Kalman filter[53]) or estimated from closed-loop BMI control (for example, the recalibrated feedback intention-trained Kalman filter[33,50]). However, to date, these approaches have 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 subsampled neurons from the complete neural population (202 neurons total) and used the subsampled 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 GLM framework[54] 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; for example, 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[41]). 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 log likelihood per spike 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.

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 (for example, 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 principal component analysis (PCA). Within this subspace (neural state space), we then used the jPCA technique to find planes that are best fit by an LDS 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 were 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 before 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 (5×10^{-5}). For the initial dimensionality-reduction step (PCA), ten principal components 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 (that is, 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, versus 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 end point (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 (ICs) vector were fixed (that is, 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 (that is, 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; that is, 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.

Figure 4—kinematic predictions of LFADS multi-session and single-session models. We used OLE 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 for the maze dataset. For Gaussian smoothing, the millisecond-binned spike trains were convolved with a Gaussian function with standard deviation 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 data sets. 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 ref. [21]. Briefly, for each of the single-session models and the multi-session model, we performed demixed PCA[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 0–1. This projection of the data we refer to as the condition independent signal (CIS), following ref. [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, although 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 demixed principal components analysis to identify the CIS dimension, as well as eight dimensions which preferentially explained condition-dependent variance (variation due to reach direction, and mixtures of reach direction and time). In this eight-dimensional condition-dependent space, we used jPCA to find a plane where trajectories exhibited rotational structure[3]. We then constructed a 3D subspace for visualization by taking the CIS dimension as well as the two dimensions comprising the first jPCA plane.

Figure 5—t-SNE visualization for cursor jump data. The pattern of inputs inferred by LFADS for individual trials was mapped into a 2D 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 under "Computing posterior averages of model variables." LFADS inferred the input values at 10-ms resolution (that is, the resolution at which the neural data were binned before being passed into LFADS). These values were then smoothed using a causal Gaussian filter with a 20-ms s.d. 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 preprocesses data by reducing its dimensionality via PCA, and the dimensionality of the preprocessed data was chosen to be 30 dimensions. The t-SNE perplexity parameter was set to 30, and sweeping this parameter between 10 and 50 had little qualitative effect on the discernibility of the three data clusters.

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, 2 kHz). Human data were 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 fourth-order Butterworth filter to minimize the contribution of action potentials to the LFP signal. Both a forward and a backward pass of the filter (that is, acausal filtering) were used to minimize group delay. Data were then filtered again with an anti-aliasing filter (eighth-order Chebyshev type I low-pass 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 LFPs 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.

Reporting summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

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 along with extensive technical documentation can be found at https://lfads.github.io.

Data availability

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

References

42. Rezende, D. J., Mohamed, S., & Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. 31st International Conference on Machine Learning* (eds Xing, E. P. & Jebara, T.) 1278–1286 (JMLR/Microtome Publishing, Brookline, MA, 2014).
43. Gregor, K., Danihelka, I., Graves, A., Rezende, D. J. & Wierstra, D. DRAW: a recurrent neural network for image generation. *arXiv* Preprint at https://arxiv.org/abs/1608.06315 (2016).
44. Krishnan, R. G., Shalit, U. & Sontag, D. Deep Kalman filters. *arXiv* Preprint at https://arxiv.org/abs/1511.05121 (2015).
45. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* Preprint at https://arxiv.org/abs/1412.3555 (2014).
46. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* Preprint at https://arxiv.org/abs/1207.0580 (2012).
47. Zaremba, W., Sutskever, I. & Vinyals, O. Recurrent neural network regularization. *arXiv* Preprint at https://arxiv.org/abs/1409.2329 (2014).
48. Bowman, S. R. et al. Generating sentences from a continuous space. In *Proc. 20th SIGNLL Conference on Computational Natural Language Learning* (eds Riezler, S. & Goldberg, Y.) 10–21 (Association for Computational Linguistics, Stroudsberg, PA, 2016).
49. Hochberg, L. R. BrainGate2: feasibility study of an intracortical neural interface system for persons with tetraplegia (BrainGate2). *ClinicalTrials.gov* https://www.clinicaltrials.gov/ct2/show/NCT00912041 (2018).
50. Gilja, V. et al. A high-performance neural prosthesis enabled by control algorithm design. *Nat. Neurosci.* **15**, 1752–1757 (2012).
51. Maaten, Lvander & Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).
52. Willett, F. R. et al. Feedback control policies employed by people using intracortical brain-computer interfaces. *J. Neural Eng.* **14**, 016001 (2017).
53. Fan, J. M. et al. Intention estimation in brain–machine interfaces. *J. Neural. Eng.* **11**, 16004 (2014).
54. Paninski, L. Maximum likelihood estimation of cascade point-process neural encoding models. *Network* **15**, 243–262 (2004).

nature research

Corresponding author(s): David Sussillo <sussillo@google.com>

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see Authors & Referees and the Editorial Policy Checklist.

Statistical parameters

When statistical analyses are reported, confirm that the following items are present in the relevant location (e.g. figure legend, table legend, main text, or Methods section).

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐	An indication of whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☐	☒	A description of all covariates tested
☒	☐	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistics including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☐	☒	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated
☐	☒	Clearly defined error bars *State explicitly what error bars represent (e.g. SD, SE, CI)*

Our web collection on statistics for biologists may be useful.

Software and code

Policy information about availability of computer code

Data collection	Custom written Matlab/Simulink routines were written to collect the data in this paper.
Data analysis	Custom written Matlab and Python / numpy routines were written to analyze the data. Further, TensorFlow was used to write the main LFADS training algorithm, which is publicly available at https://github.com/tensorflow/models/tree/master/research/lfads Additional training and data routines, as well as a documentation and a matlab interface are available at https://github.com/lfads/lfads-run-manager/

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors/reviewers upon request. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

Data from non-human primates are available upon reasonable request.
Data from human research participants cannot be made available due to patient privacy / HIPAA.

Field-specific reporting

Please select the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/authors/policies/ReportingSummary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	For the maze analyses (Fig. 2) and perturbation analyses (Fig. 5), >2000 trials were used, which far exceeds the general standards for the field. For the LFADS/GPFA decoding comparison (Fig. 2e), enough random draws of neurons were selected to ensure there was no overlap between the populations being compared. For the stitching analysis (Fig. 5), the number of sessions used (44) is far greater than is typical for the field.
Data exclusions	All exclusions were performed without regard to the outcome of the study. For the stitching analysis, datasets were excluded if the datasets did not contain enough trials to fit a model (i.e., datasets with fewer than 200 trials were excluded). For all other analyses, trials were excluded if the animal failed to complete the task.
Replication	To ensure replication, we created a comprehensive, open-source framework so that other users can apply our analyses. We have also created open-sourced code that generates standardized datasets that can be used to replicate the methods. The latter is included as part of the LFADS training algorithm, (publicly available: https://github.com/tensorflow/models/tree/master/research/lfads). The former are included with documentation and the full Matlab interface (publically available: https://github.com/lfads/lfads-run-manager/)
Randomization	For the neuron-dropping analysis (Fig. 2e,f), neurons were randomly subsampled from the full neural population without replacement using a pseudo-random number generator. Number of random sets for each population size (25, 50, 200, 150) was chosen as sufficient to verify that the populations were non-overlapping.
Blinding	All data were originally collected for studies prior to the current study. Therefore experimental design for all data collection was not influenced by the goals of the current study.

Reporting for specific materials, systems and methods

Materials & experimental systems

n/a	Involved in the study
☒	☐ Unique biological materials
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology
☐	☒ Animals and other organisms
☐	☒ Human research participants

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Animals and other organisms

Policy information about studies involving animals; ARRIVE guidelines recommended for reporting animal research

Laboratory animals	Data from 2 rhesus macaques were used in this study: Monkey J (male, 7 & 14 years old at the time of data collection) and

Laboratory animals	monkey P (male, 8 years old at the time of data collection).
Wild animals	No wild animals were used in this study.
Field-collected samples	No field-collected samples were used in this study.

Human research participants

Policy information about studies involving human research participants

Population characteristics	Data used were from two research participants with paralysis. Permission for these studies was granted by the US FDA (IDE) and Institutional Review Boards of Stanford University, Partners Healthcare/Massachusetts General Hospital, the Providence VA Medical Center, and Brown University. Participants in this study were enrolled in a pilot clinical trial of the BrainGate Neural Interface System. Informed consent, including consent to publish, was obtained from the participants prior to enrollment. Pariticipant T7 was a male, 54 years old at the time of the research collections in the reported study. Participant T5 is a male, 63 years old at the time of the research sessions reported in this study. The conclusions of this study are independent of specific characteristics of the research participant population enrolled in this study.
Recruitment	Permission for these studies was granted by the US FDA (IDE) and Institutional Review Boards of Stanford University, Partners Healthcare/Massachusetts General Hospital, the Providence VA Medical Center, and Brown University. Participants in this study were enrolled in a pilot clinical trial of the BrainGate Neural Interface System. Informed consent, including consent to publish, was obtained from the participants prior to enrollment. Recruitment of participants matched the guidelines of the trial as set forth by the FDA IDE and IRBs. Participant recruitment has no effect on any conclusions of this study.