Neuron

Viewpoint

The centrality of population-level factors to network computation is demonstrated by a versatile approach for training spiking networks

Highlights

- Latent factors enable robust, flexible training of artificial spiking neural networks
- Model networks bridge the gap between spiking- and factor-based levels of description
- Concepts such as "firing rate" become well defined in terms of factors
- Results justify the use of factors and rates in empirical hypotheses and explanations

Authors

Brian DePasquale, David Sussillo, L.F. Abbott, Mark M. Churchland

Correspondence

bddepasq@bu.edu

In brief

DePasquale et al. introduce a method for building artificial spiking neural networks using latent factors, a powerful concept in data analysis not yet grounded in neurophysiological terms. Factors are computationally central, enabling robust, flexible learning. Networks clarify that "factor" and firing rate are concretely defined and crucial to understanding computation.

DePasquale et al., 2023, Neuron *111*, 631–649
March 1, 2023
https://doi.org/10.1016/j.neuron.2022.12.007

CellPress

Neuron

CellPress

Viewpoint

The centrality of population-level factors to network computation is demonstrated by a versatile approach for training spiking networks

Brian DePasquale,[1,2,3,10,11,*] David Sussillo,[4,5] L.F. Abbott,[2,3,6,7,8] and Mark M. Churchland[2,6,8,9]
[1]Princeton Neuroscience Institute, Princeton University, Princeton NJ, USA
[2]Department of Neuroscience, Columbia University, New York, NY, USA
[3]Center for Theoretical Neuroscience, Columbia University, New York, NY, USA
[4]Department of Electrical Engineering, Stanford University, Stanford, CA, USA
[5]Wu Tsai Neurosciences Institute, Stanford University, Stanford, CA, USA
[6]Zuckerman Mind Brain Behavior Institute, Columbia University, New York, NY, USA
[7]Department of Physiology and Cellular Biophysics, Columbia University, New York, NY, USA
[8]Kavli Institute for Brain Science, Columbia University, New York, NY, USA
[9]Grossman Center for the Statistics of Mind, Columbia University, New York, NY, USA
[10]Lead contact
[11]Present address: Department of Biomedical Engineering, Boston University, Boston MA, USA
*Correspondence: bddepasq@bu.edu
https://doi.org/10.1016/j.neuron.2022.12.007

SUMMARY

Neural activity is often described in terms of population-level factors extracted from the responses of many neurons. Factors provide a lower-dimensional description with the aim of shedding light on network computations. Yet, mechanistically, computations are performed not by continuously valued factors but by interactions among neurons that spike discretely and variably. Models provide a means of bridging these levels of description. We developed a general method for training model networks of spiking neurons by leveraging factors extracted from either data or firing-rate-based networks. In addition to providing a useful model-building framework, this formalism illustrates how reliable and continuously valued factors can arise from seemingly stochastic spiking. Our framework establishes procedures for embedding this property in network models with different levels of realism. The relationship between spikes and factors in such networks provides a foundation for interpreting (and subtly redefining) commonly used quantities such as firing rates.

INTRODUCTION

Individual neurons and their spikes are the fundamental units from which a mechanistic understanding of neural computation must be built. Yet, analysis frequently abstracts away from individual spikes and focuses on a firing rate that governs the propensity to spike. A growing trend is to treat firing rates as reflecting population-level "latent factors." Experiments seek to estimate firing rates and latent factors because hypotheses employ these concepts. In sensory systems, firing rates are hypothesized to be functions of stimulus properties, and decoding operates upon rates or factors. Decisions are hypothesized to employ latent variables reflected in firing rates.[1,2] Increasingly, hypotheses regarding computation are expressed in terms of flow-fields shaping the multi-dimensional trajectory of latent factors.[3–19] Such hypotheses aim to be mechanistic and thus explanatory. For example, it is proposed that factor-level dynamics explain a causal chain in which pre-movement activity culminates in movement.[20,21]

Spike-based interactions among neurons are patently mechanistic, but do flow field arrows guiding abstracted state-space trajectories constitute a "mechanism"? The hope that they do is bolstered by the ability to train recurrent neural networks to perform brain-like computations and to understand their solutions in terms of reduced-dimensional projections of activity. Yet, the vast majority of such networks employ rate-based units and thus assume what one might hope to establish: that firing rates, and latent factors derived from them, are a valid abstraction. There is also ambiguity regarding whether rate-based units are analogous to neurons or simply an effective basis for approximating arbitrary functions and dynamics.[22,23]

If concepts such as latent-factor dynamics and firing rates are to form the foundation not just of analysis but of mechanistic understanding, it must be possible to link them concretely to individual neuron spikes. A challenge is that neither latent factor nor even firing rate have accepted physiological definitions. Except when spikes are unusually plentiful, there is no accepted procedure (biophysical or analytical) for deriving a smooth rate

from a neuron's single-trial spike train. This has prompted the concern that when spiking is intermittent (as is typical in cortex) "it does not make sense to talk about the firing rate."[24] Firing-rate-based analyses have been described as ad hoc "with virtually no empirical or theoretical support"[25] and lacking any clear link to mechanism. The notion of firing rate is not solidified by moving to a more detailed level; the membrane potential does not correspond to a rate that yields probabilistic spiking via cell-intrinsic mechanisms. Moving to a broader level is more promising; a neuron's rate might be defined as the average spiking frequency of many self-similar neurons.[26–28] Response heterogeneity (e.g., Churchland and Shenoy[29]) suggests generalizing this definition of firing rate to weighted averages of spiking activity. That approach accords with common methods for statistically estimating latent factors but leaves open the question of whether factors are a statistical convenience or a mechanistically meaningful concept.

The goal of this study is to determine whether latent-factor dynamics provide a principled way of understanding how spiking networks—specifically networks with realistic spiking variability and response heterogeneity—perform computations. We build upon studies that have successfully trained recurrent spiking networks to perform canonical computations[23,30–42] or emulate continuously valued dynamical systems.[23,26,30–32] We introduce an approach that focuses specifically on factors as training targets. Our method explicitly and concretely defines factors in terms of the synaptic and physiological elements of a spiking neural network. The focus on factors improves the flexibility and power of training; it allows construction of spiking networks that perform a wide variety of tasks, including emulating data-derived dynamics, and that perform robustly even when spiking is realistically variable.

Spiking networks trained via this approach displayed properties resembling those observed empirically, including mixed selectivity and seemingly noisy spiking. Nevertheless, factors were reliable and provided the best way to understand network function: computation was performed by factor-level interactions mediated by seemingly noisy spiking. This remained true when network realism was increased to include excitatory and inhibitory cell types and sparse connectivity. Spiking networks shared a factor-level correspondence with traditional rate-based networks, yielding a simple procedure for training spiking networks to perform computations normally instantiated within rate networks. The correspondence validates rate-based networks as a powerful tool for modeling factor-level computations, yet cautions against viewing rate units as idealized spiking neurons. For our spiking-network neurons, firing rate was not a local property but was definable only via population-level factors. These results illustrate that, although the abstraction of factors can seem uncomfortably removed from the cellular level, factors are both well defined physiologically and can be essential for understanding computational mechanism.

RESULTS

Constructing factor-based spiking models

Extracting factors from neural recordings involves analysis, often employing steps unlikely to be performed by real neurons in real time (e.g., trial averaging or statistical inference). Nevertheless, it is often proposed that computation can be modeled via factor-level interactions, or that factors provide a basis for outputs. Can the abstraction of factors be linked to physiological components within neural circuits? There exist two, hopefully compatible, perspectives regarding the relationship between spiking neurons and continuously valued factors. In one perspective (Figure 1A, left), neurons are primary, and factors summarize important aspects of spiking-neuron connectivity and activity. In the other perspective (Figure 1A, right), factors are viewed as primary, and spiking neurons instantiate a function that creates factor-level dynamics. We construct spiking networks (Figure 1B) to ask whether both views can be simultaneously valid, and whether the factor-level perspective can aid both network training and understanding of network computation.

This strategy involves a technical goal and a conceptual goal. The technical goal is to design networks that embody common assumptions about factors, accord with basic properties of neural circuits, and display spiking patterns that are realistically variable. The conceptual goal is to analyze and understand how such networks compute and ask when and whether common abstractions—such as factors and firing rates—are useful.

Networks employ leaky integrate-and-fire (LIF) neurons that spike when the membrane potential reaches threshold. Spikes impact post-synaptic membrane potentials after being filtered with two characteristic synaptic time constants, 5 and 100 ms, chosen to respect the diversity of synaptic timescales while maintaining model simplicity. The collection of filtered spike trains for all neurons and both timescales is denoted by a time-dependent vector $\boldsymbol{s}(t)$. Multiplication of $\boldsymbol{s}(t)$ by a synaptic weight matrix, $\boldsymbol{J}$, determines a vector of post-synaptic inputs, $\boldsymbol{Js}(t)$ (in units of mV; synaptic current multiplied by unity input resistance). Given two timescales and N neurons, $\boldsymbol{s}(t)$ contains $2N$ elements and $\boldsymbol{J}$ is of dimension $N\times 2N$ (N neurons with $2N$ incoming connections).

How should one identify values of $\boldsymbol{J}$ so that the network supports factors? This problem can be partitioned in two, with each half aligning with a key assumption regarding factors. First, if factors form the basis of computation (not just analysis), they must be concretely definable in terms of network elements. We make the simplest assumption: the factors the network constructs, $\boldsymbol{y}(t)$, are linear sums of single-trial synaptically filtered spiking activity:

$$\boldsymbol{y}(t) = \boldsymbol{ws}(t). \qquad \text{(Equation 1)}$$

The matrix $\boldsymbol{w}$ (Figure 1B, green) is of size $P \times 2N$, where P is the number of factors ($2N$ is twice the number of neurons due to two synaptic timescales).

The second assumption is that if spiking reflects both factors (e.g., in the average spike-rate) and signals unrelated to factors (perhaps manifested as trial-to-trial variability), each neuron's input must contain both the factors and a "non-factor-related" component. We consider that these two input components arise from two distinct connectivity components. We define $\boldsymbol{J}_{\text{fac}}$ as the component of connectivity necessary to generate the factors. Multiplying $\boldsymbol{J}_{\text{fac}}$ by the synaptically filtered spikes yields $\boldsymbol{J}_{\text{fac}}\boldsymbol{s}(t)$, the recurrent input each neuron receives via this

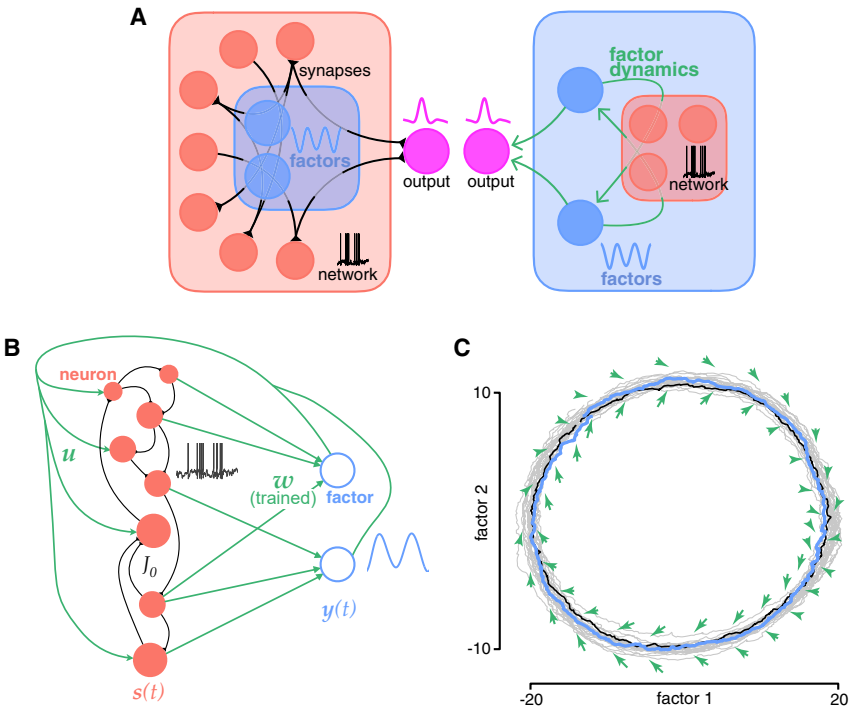

Figure 1. Factors provide a way of understanding computation in spiking networks

(A) Our spiking networks can be viewed from two complementary perspectives. Left: a neuron-centric view. Causal interactions involve connections (black synapses) among model spiking neurons (red circles). Causal interactions flow "through," and thereby construct, a modest number of factors (blue circles). Outputs (magenta) derive directly from the spiking neurons. Right: a factor-centric view. The computationally relevant interactions (green) occur among the factors (blue). Those interactions flow through a large population of spiking neurons (red). The population can instantiate a broad range of nonlinear functions, allowing a broad range of potential dynamics. Outputs are derived from the factors.

(B) Schematic of network architecture. Spiking neurons (red) interact through synapses that contain both an untrained, random aspect, $\boldsymbol{J}_0$, (black) and a trained aspect that is decomposed into two components: $\boldsymbol{w}$ and $\boldsymbol{u}$ (green). $\boldsymbol{w}$ captures the linear dependence of factors (blue circles) upon neural activity and is trained. $\boldsymbol{u}$ captures the impact of the factors back onto the neurons and is random and untrained. Factors and their associated connections are illustrated as explicit model elements but this is not necessary; the influence of $\boldsymbol{J}_0$, $\boldsymbol{w}$, and $\boldsymbol{u}$ can be combined into a unified set of connections, $\boldsymbol{J}$.

(C) Network dynamics can be understood based on factor trajectories in state space. Gray trajectories show the evolution of the first two factors for twenty trials of the cycling task. We show many overlapping gray traces to convey the overall variability of the factors. A single example trial is shown in black. The blue trace shows the mean trajectory across 100 trials. Although trajectories vary modestly across trials, factor-level dynamics are stable as illustrated by the stability of the dynamical flow field (green arrows). The flow field was determined by perturbing the factors and measuring their recovery.

connectivity component. The assumption that $\boldsymbol{J}_{\text{fac}}$ conveys factor-related input implies

$$\boldsymbol{J}_{\text{fac}}\boldsymbol{s}(t) = \boldsymbol{u}\boldsymbol{y}(t), \qquad \text{(Equation 2)}$$

where $\boldsymbol{u}$ is an $N\times P$ matrix specifying how N neurons are impacted by P factors (Figure 1B, green). Respecting the assumption that each neuron reflects random combinations of factors,[43] we pick the entries of $\boldsymbol{u}$ randomly, resulting in "mixed selectivity" (e.g., Machens et al.[44] and Rigotti et al.[45]). Although other choices are possible, this choice is advantageous because network construction becomes insensitive to the factor basis; two networks trained using different linear transformations of the same target factors will be functionally equivalent.

We hypothesize that response variability arises due to connectivity that is functionally irrelevant to the factors but supports performance (or future learning) of other computations. We define this component as $\boldsymbol{J}_0$ (Figure 1B, black). This additional connectivity is unrelated to the current factor-mediated computation and on its own should produce functionally irrelevant synaptic input. To incorporate this possibility in a simple way, we choose the entries of $\boldsymbol{J}_0$ randomly.

We define the full connectivity matrix as $\boldsymbol{J} = \boldsymbol{J}_{\text{fac}} + \boldsymbol{J}_0$. Note that $\boldsymbol{J}_{\text{fac}}$ and $\boldsymbol{J}_0$ do not reflect anatomically separate synapses—each synaptic weight in $\boldsymbol{J}$ reflects both components. Recalling that $\boldsymbol{y}(t) = \boldsymbol{w}\boldsymbol{s}(t)$, the relationship $\boldsymbol{J}_{\text{fac}}\boldsymbol{s}(t) = \boldsymbol{u}\boldsymbol{y}(t) = \boldsymbol{u}\boldsymbol{w}\boldsymbol{s}(t)$ implies $\boldsymbol{J}_{\text{fac}}$ is low rank[23,32,46] with rank equal to the number of factors. Adding $\boldsymbol{J}_0$ means $\boldsymbol{J}$ is full rank, while containing a learned low-rank component.[47]

Network training identifies $\boldsymbol{w}$ so that $\boldsymbol{y}(t) \approx \boldsymbol{y}_{\text{targ}}(t)$, where $\boldsymbol{y}_{\text{targ}}(t)$ are a set of target factors (i.e., a set of factors we are seeking to construct). We employ recursive least squares (RLS) when learning $\boldsymbol{w}$, to encourage stability of the resulting dynamics. $\boldsymbol{u}$ remains fixed and random, reflecting the choice above that individual neuron activity reflects random mixtures of the factors. $\boldsymbol{J}_0$ also remains fixed and random, reflecting the assumption that it is not tailored to the present computation. Learning $\boldsymbol{w}$ with RLS is similar to FORCE learning in Sussillo and Abbott[48] and Nicola and Clopath,[39] but with a key conceptual and practical difference. In FORCE learning, the optimization of network output creates internal signals that perform the underlying computations. We found that standard FORCE learning typically worked poorly when spiking variability was considerable and firing rate ranges were reasonable. Additionally, because we wished to explore whether and how computation can be described at the factor level, we wished factors to be explicitly defined rather than to implicitly emerge. We thus used internal training targets (i.e., factors) rather than target outputs. We chose target factors by deriving them from data or from a firing rate model. This choice ensured that target factors were sufficient to perform

the computation; ideally no additional factors would need to emerge during training. This approach relates to the full-FORCE approach[49] but uses low-dimensional training targets.

We found that the above training procedure was effective for any reasonably chosen set of target factors (i.e., factors with low trajectory tangling; see STAR Methods). Intuitively, for almost any momentary pattern of spiking, $\boldsymbol{s}(t)$, one can find a $\boldsymbol{w}$ such that $\boldsymbol{ws}(t)$ approximates the current values of $\boldsymbol{y}_{\text{targ}}(t)$. Having done so, the factor-based connectivity $\boldsymbol{J}_{\text{fac}}$ ensures that each neuron receives a factor-related input. Spiking reflects this input, improving the ability, at future moments, to find a $\boldsymbol{w}$ that extracts the factors. Initially $\boldsymbol{w}$ must be constantly updated, but with time RLS converges on a $\boldsymbol{w}$ that works well consistently. Convergence was rapid, e.g., for the cycling task considered below, training typically converged after a few hundred examples. One reason for rapid convergence is that training only had to learn to correct activity deviations that impacted the factors. It was not necessary to achieve stable trajectories in the full-dimensional space. Indeed, trained network activity followed highly variable high-dimensional trajectories (resulting in considerable spiking variability, documented below) despite adhering close to the target factor-level trajectory.

Dynamics at the level of the factors

We trained a network to generate factors extracted from empirical data, both to show the flexibility of the factor-based training approach and to explore the relationship between factor-level dynamics and the more obviously mechanistic level of spiking. We obtained target factors from the extracellularly recorded spiking activity of 109 well-isolated single neurons recorded from primary motor cortex (M1) as monkeys cycled a hand-held pedal.[14] Empirical factors were estimated using dimensionality reduction after temporal filtering and trial averaging.[14] We used principal component analysis (PCA) to reduce the population response (a $109 \times T$ matrix) to twelve factors (a $12 \times T$ matrix), with T being the number of time points during one cycle. The reduced-dimensional data define a vector, $\boldsymbol{y}_{\text{targ}}(t)$, describing the temporal evolution of twelve target factors. Each recorded neuron's trial-averaged firing rate was well approximated by a weighted sum of these factors (99% of the variance for uni-directional cycling). Using PCA was not critical; factor analysis yielded 12 factors spanning an almost identical space. Similar factors were identified without trial averaging.[50]

Our network construction procedure succeeded in training a network of 800 neurons to produce the empirical factors. Unlike for empirical data, model factors do not need to be estimated but are explicitly defined by the twelve-dimensional vector $\boldsymbol{y}(t)$, which is simply a weighted sum of spikes: $\boldsymbol{y}(t) = \boldsymbol{ws}(t)$. Plotting the first two network factors (Figure 1C; one trial shown in black, average of 100 trials in blue) yielded a circular trajectory like that seen empirically.[14] The match between network and target factors was excellent (Figure 2A, blue and dotted-black traces overlap). Spiking occurred at reasonable physiological rates (12 spikes/s on average). The functionally irrelevant connectivity component, $\boldsymbol{J}_0$, produced variable spiking (quantified further below) but did not significantly disrupt factor trajectories, which were similar across trials (Figure 1C, gray traces).

The operation of the spiking network clarifies that both the "neuron-centric" (Figure 1A, left) and "factor-centric" (Figure 1A, right) perspectives are valid. One can consider neurons (Figure 1B, red) primary, with factors mediating how current spiking influences future spiking. Alternatively, one can consider factors (Figure 1B, blue) primary, influencing their own future values via a population of spiking neurons. Factors also provide a basis for network outputs: any linear readout of the factors is a plausible network output because it is also obtainable as a linear combination of spiking-neuron activity. As we show below, the only reliable outputs are combinations of the factors. The factor-centric view further assumes that computationally relevant aspects of network dynamics are describable at the factor level, without needing to know the activity of individual spiking neurons. This was indeed true to a first approximation. A flow field supporting a stable limit cycle in factor space (Figure 1C, green arrows) is revealed by factor trajectories during recovery from perturbations. Strictly speaking, the flow field is fully defined only in the full N-dimensional space; factor-space trajectories exhibit small variations (gray traces) that cannot be explained by purely factor-level dynamics. Nevertheless, factor-based dynamics provide a good way of understanding how the network performs its function: a stable limit cycle produces repeating patterns (Figure 2A) that provide a basis for outputs.

Establishing analytically when the factor-centric view provides a good approximation, and when factor-level dynamics are stable, is difficult (e.g., Schuessler et al.[47]) and essentially impossible as network realism increases. Nevertheless, two empirical features indicate when the factor-centric view is likely appropriate. First, spiking should appear quasi-random,[37] resembling a point (e.g., Poisson) process with an underlying rate for every neuron. Second, this rate should be a function of the factors with no additional terms needed. Together, these features dictate that future factor values are approximately a weighted sum of rates, which are in turn functions of the present factor values, yielding dynamics describable at the factor level. This is an approximation because factors are actually weighted sums of spikes, not rates. However, it can be a very good approximation if the impact of spiking variability is reduced when spikes are summed across many neurons.

In the following sections, we document these features and their origins. We show that, as realism is added to the connectivity, training continues to produce dynamics that are well described at the factor level. Finally, we show that our procedure allows spiking networks to efficiently perform computations typically implemented in rate-based networks.

Reliable factors and outputs despite spiking variability

The network documented in Figures 1 and 2 was trained to produce the target factors after receiving an input pulse. Membrane voltages were randomly initialized before the pulse. Nevertheless, the network produces similar factors on every trial (Figure 2A, blue and gray traces show two example trials; 2% median single-trial error). In contrast, individual neuron membrane potentials evolve very differently across trials (Figure 2B, compare red and gray traces). Consequently, spiking is variable. On single trials, it is hard to discern a clear relationship between

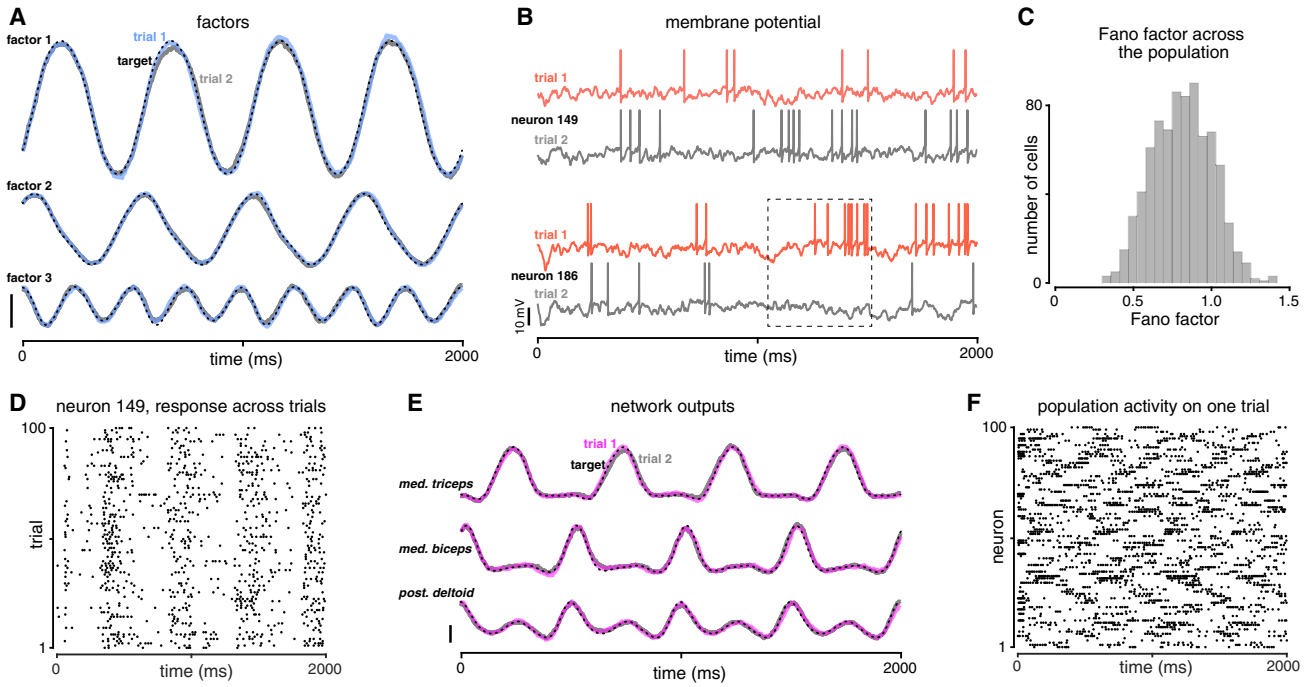

Figure 2. Spiking network model produces the empirical factors recorded during cycling

(A) The network model produces factors that are consistent from trial to trial. The first 3 factors (of 12) produced by the spiking network model on two different trials (blue and gray). The first two factors are those plotted in state space in Figure 1C. The dotted black trace plots target factors. Calibration indicates 10 arbitrary units (a.u.).

(B) Subthreshold potentials and spiking activity vary across trials. Membrane voltage is plotted for two example neurons (bottom and top) and two trials (red and gray). Dotted box highlights the second to last cycle of the task and illustrates that different numbers of spikes are emitted on different trials. The across-trial Fano factor of these two neurons was 1.10 and 0.99.

(C) Histogram of Fano factor values across the population.

(D) Individual neuron spiking varies across trials but nevertheless reflects a reliable component, such that a periodic pattern is visible when considering many trials. Raster plot of 100 randomly selected trials from one neuron (Fano factor of 0.99).

(E) The reliability of the factors allows the network to produce reliable outputs. Three network outputs are shown on two trials (magenta and gray). The target network output (recorded muscle activity) is shown in dotted black. Calibrations are 0.2 a.u.

(F) Spiking is asynchronous across the network. Raster plot of 100 randomly selected neurons for one trial.

single-neuron spiking and the factors. For example, in Figure 2B (bottom), spiking is plentiful on some cycles/trials and scant on others (compare within dotted box). For this neuron, the Fano factor was 0.99 (based on across-trial spike-count variability in 100 ms cycle-locked windows; see STAR Methods). Across neurons, the average Fano factor was 0.68 (range: 0.38–1.42; Figure 2C), in general agreement with values observed during movement.[51] Clear temporal modulation of spiking emerges only when multiple trials are considered (Figure 2D), as is commonly observed in empirical recordings.

Variable membrane-voltage trajectories are a consequence of different initial network states interacting with very complex (likely chaotic) dynamics.[52,53] This yielded seemingly stochastic spiking even though simulations were deterministic. Despite spiking variability, factors supported reliable outputs: weighted sums of model factors reliably decoded experimentally recorded muscle activity (Figure 2E; magenta and gray plot two representative trials). Importantly, because factors are weighted sums of spikes, each factor-based output is simply a weighted sum of single-neuron spikes. Decoding is reliable because the weighted sum, across neurons that spike largely asynchronously (Figure 2F; Figure S1), reduces variability through averaging.

Reliable aspects of neural activity reflect the factors

Empirical analyses often assume that each neuron's response reflects both reliable signals that are computationally meaningful and additional noise-like components. Network construction was inspired by this idealization, and networks did indeed exhibit spiking that was task-modulated yet also variable (Figures 2B–2D). Because model neurons were deterministic, if distinct reliable and noise-like signals impact spiking they must be present within a neuron's input.

Across neurons, 96% of the variance in the trial-averaged input was accounted for by regressing against the twelve target factors (Figure S2A). Thus, the reliable component of each neuron's response does indeed derive from the computationally meaningful factors. Of course, to be computationally meaningful, the reliable component must be present and sizable on single trials. This was indeed the case: the top twelve principle components (PCs) captured considerable variance (53%, Figure S2B) that was reliably task-related (i.e., repeated at the cycling period,

Figure S2C). If reliable signals are computationally meaningful, removing them should impair computation. This was also true: removing the neural inputs due to any of the largest PCs rendered the network non-functional (muscle activity could not be read out).

Single-trial inputs also contained a sizable unreliable component: the remaining 47% of the variance was split across many PCs (Figure S2B), displayed a broad frequency spectrum (Figure S2C), and was nearly uncorrelated across trials (average $r = 0.007$). The signals captured by these PCs were not computationally meaningful: removing the contribution of higher-order PCs had almost no impact on network outputs or factors. This was true even if we removed the contribution of hundreds of higher-order PCs, accounting for >15% of the total input variance (Figure S2D).

These findings thus support the common assumption that each neuron's response reflects both reliable and unreliable signals. The finding that the computationally meaningful component arose solely from the factors supports the validity of the factor-centric view. As a technical aside, the ability to determine that the reliable input component depended on the factors (Figure S2A) was aided by intentionally choosing a "complete" set of factors as training targets. Thus, all network factors were known; no new "emergent" factors arose. This was typical in our simulations, but exceptions are possible. Networks are nonlinear. Thus, for any given factor (e.g., a 2 Hz sine-wave) nonlinear distortions (e.g., higher harmonics) can yield "new" factors. In this case, this was irrelevant: all higher harmonics existed within the target factors so no novel factors emerged. For more complex tasks, assembling a complete set of factors may be challenging. Below, we show how this can be accomplished by employing trained rate-unit networks. We found that this procedure yields target factors sufficient to perform the computation and rich enough to contain any additional factor likely to emerge. Not only is training most likely to be successful when using a complete set of target factors,[39,49] interpretation is also simplest. For example, in the next section we leverage the explicitly defined factors to determine the origin of the reliable and unreliable synaptic-input components.

Reliable and variable synaptic-input components

Above we employed PCA, much as an experimenter might, and observed that each neuron's synaptic input displays a reliable, factor-related component and an unreliable component. Ideally one would define these components not by analysis but mechanistically, in terms of model elements. Our model-construction approach makes this easy because factors are concretely defined. Every neuron's synaptic input can be decomposed into a factor-based component and a non-factor-based component. We can write the factor-based input for neuron n as a weighted sum of factors:

$$z_n^F(t) = \sum_p^P u'_{np} y_p(t), \qquad \text{(Equation 3)}$$

where P is the number of factors. The weights $\boldsymbol{u}'$ reflect both the impact of $\boldsymbol{u}$ (which along with $\boldsymbol{w}$ defines $\boldsymbol{J}_{\text{fac}}$) and the fact that, once training is complete, factor-based information also flows through $\boldsymbol{J}_0$ (once neurons reflect factors they inevitably transmit factor-related signals through both connection components, see STAR Methods). The non-factor-based input, which arises solely from $\boldsymbol{J}_0$, is the total synaptic input minus the factor-based input.

Figure 3A shows factor-based and non-factor-based inputs (on the same scale) for one neuron and two trials. The factor-based component is similar on both trials (blue and gray traces overlap). This is true across all neurons—plotting the factor-based component on the second versus first trial reveals a high correlation (Figure 3B, blue, one point per time/neuron). This was true across all pairs of trials (average $r = 0.95$). This reliability is a direct consequence of the factors being reliable. In contrast, the non-factor-based component is variable across trials (Figure 3A, red and gray traces). This was true across all neurons (Figure 3B red) and pairs of trials (average $r = 0.024$).

A common assumption is that "meaningful" aspects of each neuron's response vary across nominally identical trials, just much less than spiking might suggest.[54] To assess whether factor- and non-factor-based inputs provide insight into this assumption, we computed the across-trial variance of each input component. Variance was computed separately for each time, averaged across time, and normalized by within-trial modulation (assessed as the across-time variance, averaged across trials). The factor-based component displayed variances above zero but well below unity (Figure 3C, blue; mean: 0.07). In contrast, values for the non-factor-based component clustered near unity (Figure 3C, red; mean 0.99). Unity values indicate that across-trial variance is as large as within-trial variance, as would be expected for pure noise. Thus, each neuron's spiking is driven by a factor-based component that is reliable (but shows modest variability; e.g., Figure 1C, gray traces) and a non-factor-based component that effectively injects noise.

These properties are consistent with a view of spiking in which computationally irrelevant variability is "layered" on top of computationally impactful variability. Yet, it is not the case that there exist "factor-based spikes" and non-factor-based spikes. Every spike is a joint consequence of both inputs. Which raises the question: why does the unreliable non-factor-based input not have a greater impact on factor reliability? The answer relates to the non-factor-based component being weakly correlated across neurons, even those with similar "tuning." Figure 3D shows factor-based and non-factor-based components for two neurons on one trial. These neurons had strongly correlated factor-based components (blue; $r = 0.97$ across all trials), but there was almost no correlation between their non-factor-based components ($r = -0.003$ for this trial, $r = 0.09$ across all trials, Figure 3E).

Consistent with the example in Figures 3D and 3E, neurons with correlated factor-based inputs did not have correlated non-factor-based inputs. Across neuron pairs, the correlation between factor-based components occasionally became large (99$^{\text{th}}$ percentiles: −0.90 and +0.90) but the correlation between non-factor-based components never did (99$^{\text{th}}$ percentiles: −0.10 and +0.11). There was essentially no relationship between these two correlations ($R^2 = 0.003$, slope = 0.005). Consistent with weak pairwise correlations, the non-factor-based component was high dimensional across neurons (Figure 3F, red).

These observations explain why a component of spiking variability appears "private." Even when two neurons share similar factor-based responses, spiking is also driven by an uncorrelated

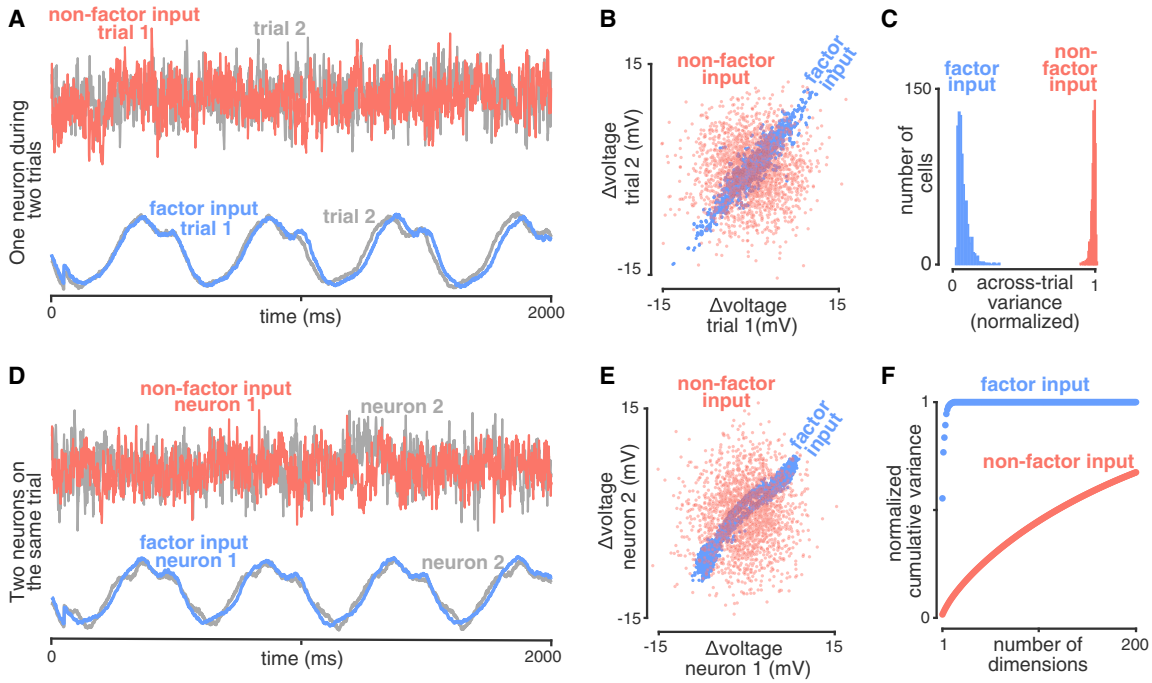

Figure 3. Synaptic inputs have reliable and variable components

The input to each neuron is partitioned into a "factor-based synaptic input" and a "non-factor-based synaptic input."

(A) Example illustrating that only the factor-based input is reliable across trials. The factor-based input (bottom) and non-factor-based input (top) are shown for one neuron and two trials (same neuron and trials for both). Both input components are plotted on the same scale.

(B) Scatterplot confirming that the finding illustrated in (A) held across all neurons. This analysis asks whether, across neurons and times, the value of the factor-based input on trial 1 is predictive of the value on trial 2. Indeed, this correlation was strong (blue, $r = 0.95$). In contrast, the correlation was very weak for the non-factor-based input (red, $r = 0.024$). Each point corresponds to one neuron and time for the same two trials as in (A).

(C) Distributions confirming that the finding in (A) and (B) held across all trials: the non-factor-based input (red) is essentially unrelated across trials while the factor-based input (blue) displays only modest across-trial variability. For each neuron, we computed the across-trial variance of the relevant input across 100 trials, then normalized by within-trial across-time variance.

(D) Example illustrating that only the factor-based input is reliable across neurons. The factor-based input (bottom) and non-factor-based input (top) are shown for two neurons and one trial (same neurons and trial for both). These two neurons were chosen because they have strongly correlated factor-based inputs ($r = 0.97$ on this trial). Nevertheless, the non-factor-based input was essentially uncorrelated ($r = -0.003$ on this trial). Both input components are plotted on the same scale.

(E) Scatterplot confirming that the finding illustrated in (D) held across all (100) trials for these two neurons: the factor-based input was strongly correlated (blue, $r = 0.97$), yet the non-factor-based input was not (red, $r = 0.09$). Analysis considers data from all trials for the same neurons as in (D). Each point corresponds to one time during one trial.

(F) Dimensionality (across neurons) of the factor-based and non-factor-based inputs. Cumulative variance accounted for is plotted as a function of the number of PC dimensions. PCs were computed from the neuron-neuron covariance matrix. Across the population, factor-based inputs were accounted for by a small number of PCs, indicating low dimensionality and strong correlations across neurons. In contrast, non-factor-based inputs were very high dimensional and thus weakly correlated among neurons.

component. Precisely because it is uncorrelated, it minimally impacts the factors. Networks thus display the remarkable property of being unreliable at the spiking level while exhibiting robust factor-level dynamics. The fact that networks can compute in this regime is relevant to many common analyses, including trial averaging and methods that infer single-trial firing rates. These methods assume spiking "noise" should be isolated from computationally important signals. This assumption is difficult to justify from a single-neuron perspective but makes perfect sense for spiking networks operating in the regime documented here.

A conceptually grounded firing rate

Analysis and interpretation often assume that a neuron's spikes probabilistically reflect an underlying firing rate, typically modeled statistically as a Poisson process with rate $r(t)$. Historically, trial averaging was used to estimate $r(t)$. Recent approaches consider $r(t)$ to be defined on single trials, and seminal studies have linked that rate to task-relevant inputs (e.g., Park et al.[55]) or internal variables (e.g., Churchland et al.[56]). Statistical analysis methods have built on this framework and modeled single-trial rates as functions of underlying factors (e.g., Pandarinath et al.[11] and Yu et al.[57]) or the activity of other neurons (e.g., Yates et al.[58]).

These analysis approaches identify quantities that, in practice, appear deeply relevant to behavior and computation. Yet, we lack a mechanistic grounding of the term firing rate. If $r(t)$ is a hidden variable that governs the probability of spiking, what and where is that variable? Are the operations used to estimate $r(t)$ on single trials purely statistical, or do they have mechanistic

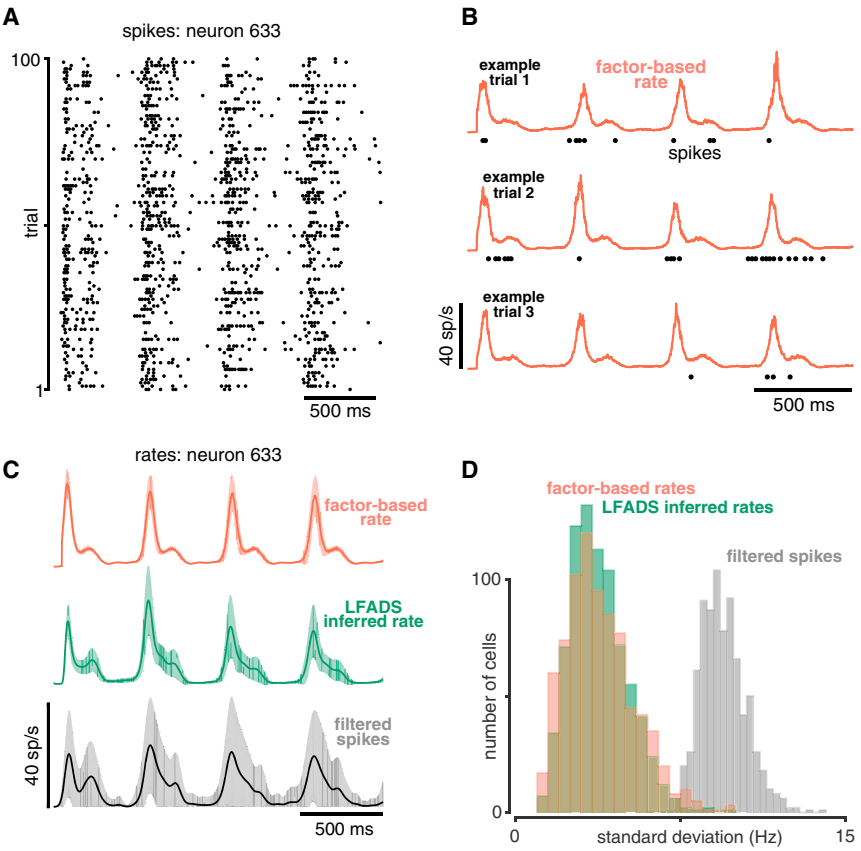

Figure 4. A conceptually grounded firing rate

(A) Spike trains from 100 trials for a single model neuron during the cycling task. Spiking was variable across trials (Fano factor of 1.37), but rhythmically structured activity becomes visible when multiple trials are observed.

(B) Our factor-based single-trial firing rate, $r_n^F(t)$, for three trials for the neuron shown in (A). Spike times on each trial are shown as dots. $r_n^F(t)$ is reliable from trial to trial, even in moments when the neuron does not spike (e.g., 0–1,000 ms in example trial 3).

(C) Comparison of firing rates computed mechanistically (factor-based), statistically by leveraging the full population (LFADS), and traditionally via filtering the neuron's spike train (20 ms Gaussian). Analysis is for the same neuron as in (A). All three methods result in similar across-trial mean firing rates (solid line) but with different degrees of trial-to-trial variability (across-trial standard deviation shown in the shaded region).

(D) Distribution (across the population) of the temporally averaged across-trial standard deviation of $r_n^F(t)$ (red), the LFADS-derived single-trial firing rate (green), and temporally filtered spike trains (gray).

meaning? For example, when estimating $r(t)$ for a given neuron, many methods leverage the spikes of other neurons (e.g., Pandarinath et al.[11] and Yates et al.[58]). Is this simply effective inference that leverages a tendency for rates to be correlated, or should a mechanistic definition of $r(t)$ incorporate population-level spiking?

A mechanistic definition of $r(t)$ can be derived from the factor-based component of a neuron's synaptic input. To do so, we respect the central idea of a rate: its goal is not to indicate with certainty whether a spike will occur (for that, one would look to the membrane potential) but to describe the probability of spiking based on computationally meaningful quantities. We define the "factor-based" firing rate of neuron n as

$$r_n^F(t) = F\left(z_n^F(t)\right), \qquad \text{(Equation 4)}$$

where $F()$ captures a monotonic, nonlinear relationship between the factor-based synaptic input $z_n^F(t)$ and the probability of spiking.

In a spiking network with any degree of realism, identifying the precise form of $F()$ is challenging. Yet, given that $F()$ must be positive-valued and monotonically increasing, it can be approximated. For simplicity, and to mesh with prior statistical approaches, we chose $F()$ to be an exponential with a neuron-specific gain and offset:

$$r_n^F(t) = \exp\left(b_n^1 z_n^F(t) + b_n^0\right). \qquad \text{(Equation 5)}$$

One could employ a more sophisticated function-class, but this would have minimal quantitative impact. For neurons with reasonable physiological rates, the exponential provides an excellent approximation (Figure S3; STAR Methods).

Defined this way, $r_n^F(t)$ displays properties typically attributed to single-trial firing rates. To illustrate, we consider one example simulated neuron. Despite variable spiking (Figure 4A), $r_n^F(t)$ is similar across trials (Figure 4B). This agrees with the common assumption, noted above, that underlying rates are more reliable than spiking would suggest. Indeed, there are moments where the neuron exhibits zero spikes over an approximately 1-s interval (Figure 4B, bottom, first two cycles). From a literalist's perspective, this would indicate a zero firing rate. Yet $r_n^F(t)$ on that trial resembles itself on other trials, consistent with $r_n^F(t)$ determining the probability of spikes that may or may not actually be produced. This relates to an important feature of $r_n^F(t)$: it can change on timescales faster than those of spiking. There is no paradox in a firing rate that modulates at 10 Hz but peaks at 5 spikes/s.

We propose $r_n^F(t)$ as a mechanistic "ground-truth" definition of a single-trial firing rate—one that can be computed when all network variables are known. Does this definition accord with statistical approaches that infer firing rates from data? This was indeed the case. We first considered the common practice of filtering (20-ms Gaussian kernel) and trial averaging single-neuron spike trains. This produced an estimated average firing rate (Figure 4C, black) that resembled the trial-averaged $r_n^F(t)$ (Figure 4C, red) but was somewhat over-smoothed. Over-smoothing could have been combated with a narrower filter, but at the cost of compromising already unreliable single-trial estimates (compare distributions in Figure 4D and envelopes in

Figure 4C). This highlights a well-known trade-off when estimating a neuron's rate using only its own spikes: no single filter choice can achieve undistorted trial averages without overestimating single-trial variability. We next considered latent factor analysis via dynamical systems (LFADS) (Pandarinath et al.[11]; see STAR Methods), which estimates single-trial rates using statistical assumptions that map well onto our network properties: factor-level dynamics, factor-based rates, and roughly Poisson spiking. LFADS-inferred rates (Figure 4C, green) closely matched $r_n^F(t)$ on average and also on single trials ($r = 0.87$). Furthermore, LFADS-inferred rates displayed across-trial variability of the same order as $r_n^F(t)$ and much less than filtered spike trains (Figure 4D).

LFADS is a particularly appropriate analysis tool, but other single-trial-focused methods would have similarly succeeded in estimating single-trial rates.[11,57,59–64] All these methods assume the spikes of the full population are informative regarding the rate of a particular neuron. Explicitly defining $r_n^F(t)$ clarifies that this assumption is fundamental. Unlike other assumptions (e.g., temporal smoothness), this is not a statistical regularity that tends to be true but emerges from the definition of $r_n^F(t)$. One cannot isolate a neuron's factor-based input component from local information; every synapse conveys both factor- and non-factor-based components. Thus, although $r_n^F(t)$ describes a property of a specific neuron (its probability of emitting a spike) it remains a population-level quantity.

Both our definition of $r_n^F(t)$ and the operation of modern methods such as LFADS suggest a conceptual reorientation. Because it has long been possible to estimate firing rates via trial averaging, the concept of firing rate is familiar and seems fundamental. This makes it tempting to view factors as an abstraction defined in terms of rates. For example, it is common to consider a high-dimensional firing rate space, with factors relating to a subspace capturing most firing rate variance.[65,66] This description holds for our model networks. Yet, thinking of factors as "low-dimensional summaries" of rates is not quite correct. Rates are defined in terms of factors, which are in fact more fundamental.

Extending to more realistic network models

The network analyzed above employs a single-model cell type forming both excitatory and inhibitory synapses. To explore whether key properties hold when additional realism is incorporated, we considered two important features of biological networks. First, because biological neurons are either excitatory or inhibitory, the columns of $\boldsymbol{J}$ should be sign-constrained (≥ 0 for excitation, ≤ 0 for inhibition). Second, in biological networks, a given neuron does not connect to all other neurons. Enforcing sparseness means $\boldsymbol{J}_{\text{fac}}$ can no longer be low rank; the probability of finding a low-rank factorization such that its outer product respects the desired sparsity pattern becomes vanishingly small as sparsity increases. It is thus an open question whether more realistic networks will continue to support a small number of factors (but see Herbert and Ostojic[67]).

We used the cycling factors as training targets for a network with sparsely connected excitatory and inhibitory neurons. $\boldsymbol{J}_0$ obeyed these constraints but was otherwise random. For technical reasons (see final paragraph of this section) a two-step procedure determined factor-related connectivity. First, we trained an unconstrained $\boldsymbol{J}_{\text{fac}}$ matrix using RLS. We then used the activity of the resulting network to train a second network with constrained factor connections, $\boldsymbol{J}_{\text{fac}}^{\text{C}}$ (STAR Methods). As a technical aside, our training approach did not allow enforcement of the same non-zero elements in both $\boldsymbol{J}_0$ and $\boldsymbol{J}_{\text{fac}}^{\text{C}}$; although $\boldsymbol{J}_0$ and $\boldsymbol{J}_{\text{fac}}^{\text{C}}$ each have 40% non-zero elements (Figure 5C) $\boldsymbol{J}$ was modestly less sparse (60% non-zero elements). This is acceptable because it is still true that $\boldsymbol{J}_{\text{fac}}^{\text{C}}$, the trained component responsible for producing the factors, no longer has the overly idealized properties of being low rank, fully connected, and unconstrained in sign.

Despite the added constraints, training was successful. The constrained network produced the target factors (Figure 5A) and outputs (Figure 5B) just as the unconstrained network did. Output error increased modestly relative to the unconstrained network (9% rather than 2% median normalized error), a consequence of fewer learnable connections that could have been counteracted by adding more neurons. The constrained network shared the previously discussed properties of the unconstrained network. For example, the twelve factors account for 97% of the variance of each neuron's trial-averaged post-synaptic input. Thus, the utility of factors as a training tool and as an abstraction does not depend on overly idealized assumptions regarding connectivity.

The two-step training method reflects the importance of training recurrent networks—sparsely or densely connected—with RLS. RLS aids stability by ensuring that, during training, the network experiences a distribution of errors (differences between the target factors and the network-produced factors) approximating what will be produced by the network post-training. Direct use of RLS was not feasible for the constrained weight matrix. Our two-step method overcomes this technical limitation by exploring likely experienced error states during initial RLS training and using samples of these states during non-RLS learning, for which constraints are more easily enforced. This approach can confer advantages over non-recursive training approaches that do not ensure the range of likely-to-be-produced errors is experienced during training. To illustrate, we examined the stability-inducing errors produced during RLS training. Errors were low dimensional (Figure 5D black), and their average auto-correlation function (Figure 5E black) revealed large temporally lagged correlations. Neither property was exhibited if errors were shuffled and thus rendered Gaussian (Figures 5D and 5E gray), the assumed distribution of errors during non-RLS learning.[23,68] As expected, when shuffled errors were used during the second training step (leading to effectively i.i.d. Gaussian perturbations off the desired network states), performance dropped significantly (median test error of 76%).

Building spiking models with model-derived factors

Training recurrent networks of continuously valued firing rate units has become a common approach for understanding brain-like computations.[48,49,69–72] Rate networks presuppose that continuously valued firing rates are the building blocks of computation, leaving unanswered the question of whether and how solutions in these networks can be realized in spiking

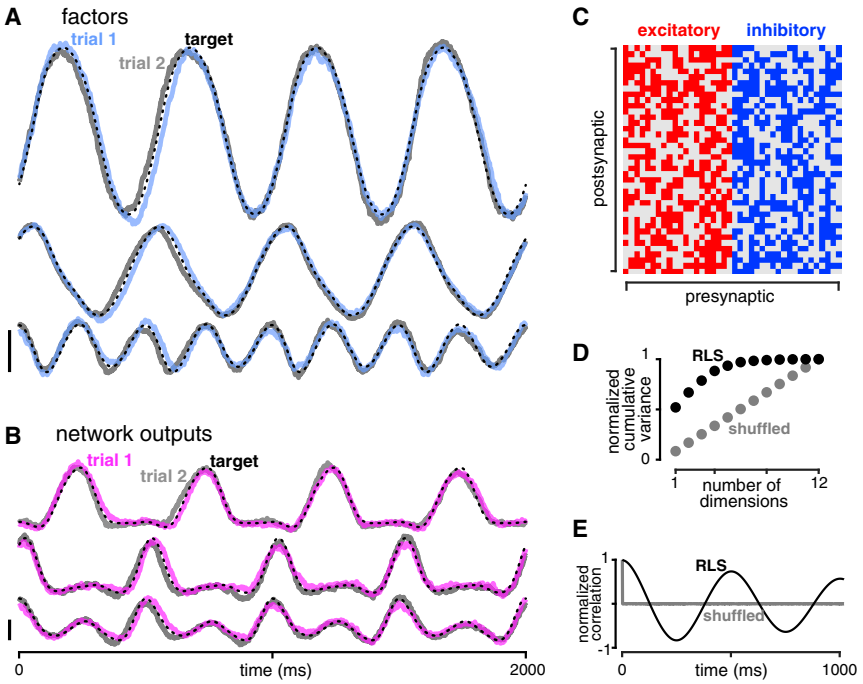

Figure 5. Extending training to more realistic network models

(A) Factors produced by a network model with constrained connectivity, trained to produce the cycling factors. The first three factors are shown for two trials (blue and gray). Factors are similar to those exhibited by an unconstrained model (compare with Figure 2A). Target factors are shown in dotted black. Calibrations are 10 a.u.

(B) Plot of three network outputs (same analysis as for the unconstrained network in Figure 2E). Target muscle activity is shown in dotted black. Calibrations are 0.2 a.u.

(C) Sign of the entries of $\boldsymbol{J}^{C}_{fac}$ for 20 excitatory neurons (red) and 20 inhibitory neurons (blue). Gray indicates no connection.

(D) Cumulative variance, captured by successive dimensions, for the error between network-generated factors and target factors during RLS learning (black) and when errors are shuffled (gray).

(E) Average temporal auto-correlation of the error between the network-generated factors and target factors during RLS learning (black) and when those errors are shuffled (gray).

networks. Rate-network computations can often be understood by examining projections of rates onto modest numbers of dimensions, using PCA or related methods.[9,15,21] These projections are weighted sums of rate-unit activity and thus are effectively factors, introducing the prospect that rate models can serve as sources of factors when training spiking models. A factor-level correspondence could provide an alternative to creating rate-based and spike-based models that correspond at the unit level.[26,27,38] We illustrate an efficient and simple means of constructing spike-based models to perform tasks using the same factor-based solution employed by a rate network. This correspondence validates the widespread use of rate-based models—although their individual elements may be unrealistic, their factor-level solutions are readily shared with spiking networks—and opens the door to building spiking networks for a wide variety of tasks. We use two tasks and two rate-network training approaches to show that training success is not sensitive to the task or rate-based training method.

Reaching task

We consider the task of generating muscle activity during reaching, a standard task for evaluating how rate-based networks transform static inputs into temporally structured outputs.[21,26,73] We considered a version of this task where a two-dimensional input (Figure 6A, blue) specified reach identity (Figure 6A insert), and termination of a third input (Figure 6A, gray) indicated that movement should begin. Accurate performance required network outputs to match the activity of two muscles (lateral biceps and posterior deltoid; Figure 6B) recorded during a center-out reaching task.[74,75]

We first used FORCE learning[48,49] to train a rate-based network (see STAR Methods for an alteration to FORCE that allowed us to sidestep extensive rate-model optimization). We reduced rate-network activity to a handful of factors by projecting onto the top PCs (computed using data from all $C = 8$ conditions; see STAR Methods). The majority (99%) of firing rate variance was accounted for by 37 factors, which were target factors for a 1,200-neuron spiking network. $\boldsymbol{J}_{fac}$ was optimized using RLS so that each spiking-network factor (Figure 6F, solid lines) matched one target factor (Figure 6F, lighter underlying traces). Training was successful; for each reach condition, the external inputs prompted generation of the target factors, which provided a basis for muscle activity (Figure 6C). Muscle activity was generated with a median error of 8% (relative to the empirical targets) similar to rate-network performance.[21]

Model spiking-neuron responses resembled those recorded from motor and premotor cortex; they were directional,[76] but tuning differed between the preparatory epoch (before the go cue) and the movement epoch (after the go cue).[77,78] For example, the neuron in Figure 6D (activity shown for 15 trials per direction) produces preparatory-epoch spikes that are most prevalent before leftward reaches (blue) and movement-epoch spikes that are most prevalent during downward reaches (yellow). The neuron's trial-averaged firing rate (Figure 6E) reveals responses whose amplitude, phase, and time-course all vary with reach direction. Such features are typical and agree with well-documented features of real neurons[20,29] and rate networks trained on similar tasks.[21,26,73] This validates the rate-modeling approach while also allowing exploration of biologically relevant features (e.g., spiking variability) absent in rate networks.

As illustrated in Figure 6D, spiking is variable across trials (average Fano factor of 0.69 across all network neurons). Factors are more consistent but display modest variability that meaningfully impacts behavior. The largest 11 factors (contributing 95% of all target-factor variance) are most responsible for network

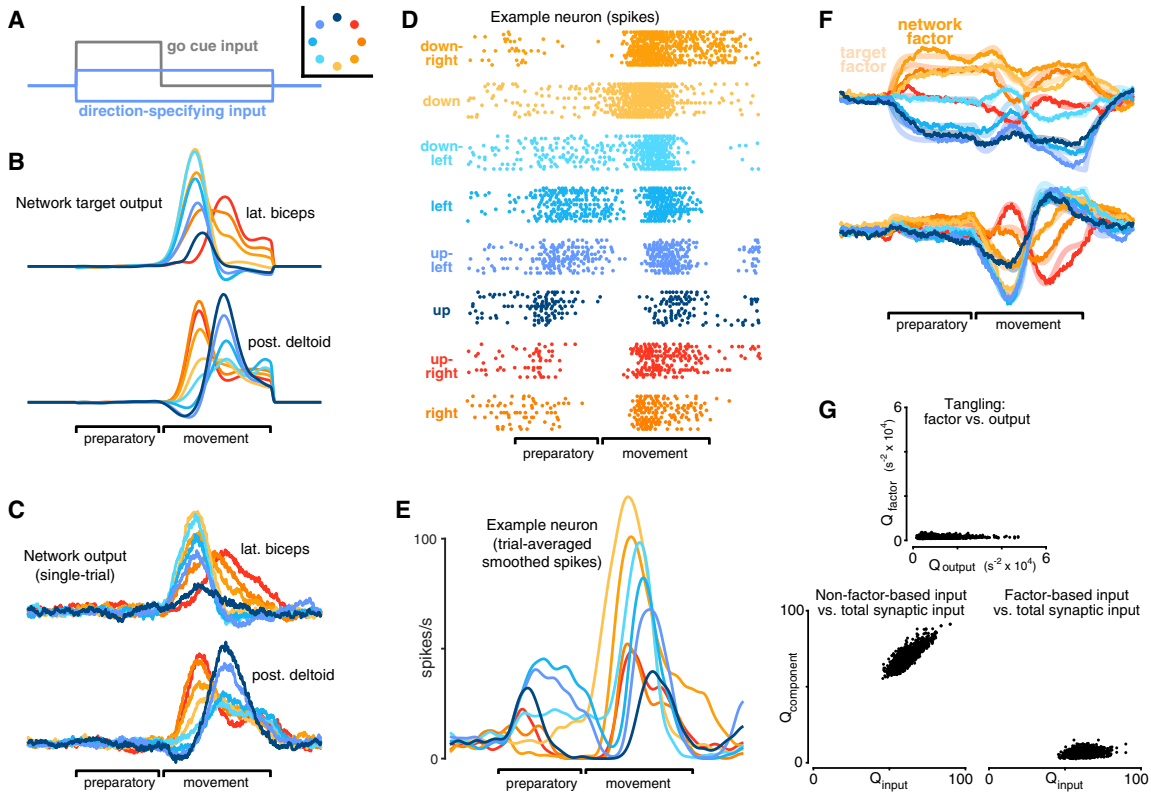

Figure 6. A reaching task

(A) Network inputs. A two-dimensional input (blue), conveying the sine and cosine of reach direction ("up-left" in this example), is applied throughout each trial. Another signal (gray) terminates at the end of the preparatory period, yielding the "go cue" that movement should begin. Inset indicates the color corresponding to each reach direction, used to plot data elsewhere in this figure.

(B) Target outputs were based on the empirically recorded EMG from two arm muscles (lateral biceps and posterior deltoid).

(C) Network outputs for both muscles, for one trial for each reach direction.

(D) Raster plot showing spikes for one model neuron for 15 trials per direction. The Fano factor for this neuron was 0.56.

(E) Trial-averaged smoothed spikes for the same neuron. Smoothing used a 20-ms Gaussian kernel.

(F) Temporal evolution of factor five (bottom) and six (top) for all reach directions on one trial. Solid (slightly noisy) lines show the factors produced by the network. Lighter underlying traces show the target factors, derived from a firing rate model.

(G) Trajectory tangling (Q) for different aspects of network activity. Top: tangling of spiking network factors versus tangling of spiking network outputs. Data were minimally smoothed (5-ms Gaussian) to emulate prior studies. Bottom-left: tangling of the non-factor-based synaptic inputs versus tangling of the total synaptic inputs. Bottom-right: tangling of the factor-based synaptic inputs versus tangling of the total synaptic inputs. All tangling analyses are applied to single-trial trajectories.

output. Their average across-trial variance (normalized by average within-trial modulation as described above) was 0.11, yielding modestly variable muscle commands (normalized variance of 0.08). This observation relates to the common analysis assumption that spiking variability has two components—rate/factor-based variability and private variability[11,79]—that can be modeled as a doubly stochastic process.[56,80] In this statistical framework, rate/factor-based variability impacts network computation and output while private variability does not. We stress that although this dichotomy is useful, it is not mechanistically accurate for two reasons. First, as discussed above, the source of across-trial spiking variability is not a private noise source but a network-derived non-factor-based input. Second, because every neuron receives both factor-based and non-factor-based inputs, non-factor-based variability will always create some factor-based variability, thereby indirectly impacting outgoing commands (although this effect may be small for large networks).

Even though the network generates outputs with non-negligible variability, it still does so robustly; spiking variability does not cause the network to become unstable. We recently argued that recurrent networks can robustly generate outputs when factors display low trajectory tangling, defined as avoiding situations where similar states have different derivatives.[14] Low tangling is most clearly necessary when dynamics are autonomous but is beneficial whenever strong dynamics are needed. Muscle trajectories are typically tangled, suggesting that trajectory tangling should be much lower for factors than for outputs.

We computed Q_{factor}, the trajectory tangling of the factors during the reaching period[14] and compared with Q_{output}, the trajectory tangling of the outputs. Both were computed for multiple times within all reach conditions, resulting in a scatterplot of

many comparisons (Figure 6G, top). Although Q_{output} often became high, Q_{factor} was consistently low. Low factor-trajectory tangling explains why the spiking network was able to learn the factors, and why spiking variability did not prevent it from robustly producing outputs. Low tangling also highlights that although factors are explicitly trained (much like outputs are traditionally trained) factors are not arbitrary network outputs. Factors should support the internal computation that produces the output—here, they are derived from a rate network that performs the task well. Prior work has hinted at this need. Nicola and Clopath[39] included additional outputs to aid training and Kim and Chow[37] stressed the utility of heterogeneous firing rates as targets. Both approaches decrease the likelihood of very high tangling. The approach of Alemi et al.,[31] Boerlin et al.,[32] and Eliasmith[23] ensured a low-tangled dynamical system by explicitly defining that system. The use of a trained rate network provides a more general solution that can work for any task for which a rate network can be trained. If needed, steps can be taken to ensure the rate-network solution is robust and has low tangling (e.g., Sussillo et al.[21] and DePasquale et al.[49]).

As documented above, networks exhibit reliable factors despite variable membrane potentials and unreliable spiking. Our modeling framework allowed us to examine the source of this potential paradox using the tangling analysis. Tangling of the full 1,200-dimensional state-trajectory (describing each neuron's synaptic input) often became high. This was not due to high dimensionality per se, but to the variable trajectories caused by non-factor-based inputs. In agreement, tangling of the non-factor-based input was high (Figure 6G, bottom-left) and correlated with that of the total input ($r = 0.82$). Tangling of the factor-based input was low relative to the tangling of the total input (Figure 6G, bottom-right). This disparity explains how networks can simultaneously exhibit reliable factor-level dynamics and unreliable spiking-level dynamics.

Contextual integration task

Rate-based models trained to perform a contextual integration task have illustrated how networks can make decisions flexibly, and network solutions have been understood in reduced-dimensional spaces.[9,46] We considered a version of this task where two sensory inputs are simultaneously presented and the network must compute the cumulative sum of only one (i.e., must "pay attention" to and integrate only one stimulus). A context-cue input, maintained throughout the trial, indicates which sensory input should be integrated. Network performance is assessed by evaluating the sign of the output at the trial's end. Sensory inputs are a series of Gaussian-distributed random numbers with fixed variance and a different mean on each trial. Because the goal is to report whether the mean is positive or negative, the magnitude of the mean determines trial difficulty. We refer to trials in which the mean was positive as "right-choice trials" and where the mean was negative as "left-choice trials" to reflect the standard structure of the empirical task. We refer to the two contexts (arbitrarily) as "red" and "blue."

We trained a spiking network using RLS, using target factors obtained by training a rate network using backpropagation through time.[38,81] We could have used FORCE (as above), but FORCE and backpropagation find noticeably different solutions. We wished to examine whether factors produced by backpropagation—a common and powerful technique—can be instantiated in a spiking network. The fact that they could (see below) does not prove this will always be feasible but opens the door to constructing spiking networks to perform a great variety of tasks.

Training was successful; a network of 800 spiking neurons performed this task. An example "blue-context" trial is shown in Figure 7A. Network output (black) correctly integrated the cued sensory input (true integral in blue) but not the uncued input (true integral in red). Performance approached 100% when the cued sensory input strongly indicated the correct choice (Figure 7B, blue). As expected, network performance was near chance (50%) when the cued sensory input had a zero mean (i.e., the trial was ambiguous). The network successfully ignored the uncued sensory input; performance was only weakly impacted by its mean (Figure 7B, red).

Spiking network performance was reduced relative to the rate model used to train it (Figure 7B, gray). This was expected both because of the decreased fidelity of spiking neurons (outputs are binary rather than continuous) and because of spiking variability. As discussed above, most spiking variability does not impact the factors. Yet, some does, especially in modest-sized networks. RLS can produce factor-level dynamics that are stable and resist this impact. Yet, continuous integration is particularly sensitive because small amounts of noise are integrated over time. Whether the impact of spiking variability on factor-level computations is significant for biological networks remains unclear; in large networks, other sources of variability may be limiting.

Single spiking-model neurons showed a variety of response features, including mixed stimulus and context selectivity, consistent with empirical findings[3,9] and rate models.[9] To illustrate, we leverage the ability to define single-trial firing rates. Single-trial rates were computed based on the factor-based input (see above). As a minor technical point, the current network receives a time-varying input that is effectively an "inherited" factor and is thus included in the factor-based input (see STAR Methods). We plot $r_n^F(t)$ of three neurons for the same four example trials (Figure 7C). One neuron (top) primarily reflects choice (responding during leftward choices). Another (middle) primarily reflects context (responding during the red context). The third neuron (bottom) shows mixed selectivity, which was typical. Context selectivity and choice selectivity were present to varying degrees in different neurons with negligible correlation (Figure 7D).

By construction, the computation was subserved by nine factors, but considering the first two is sufficient to understand key aspects of the computation. The first two factors can be represented in two-dimensional space (Figure 7E). Each point indicates the average state across many trials/time points that shared both context and similar values of factor two. Color indicates context. Intensity reflects the true average integral of the cued sensory input, and thus the evidence for the correct choice. While mixed selectivity was typical at the single-neuron level, the dominant factors (fortuitously) contain separable task-critical signals: choice is available to be "read out" using factor 2, while context has a strong impact on factor 1.

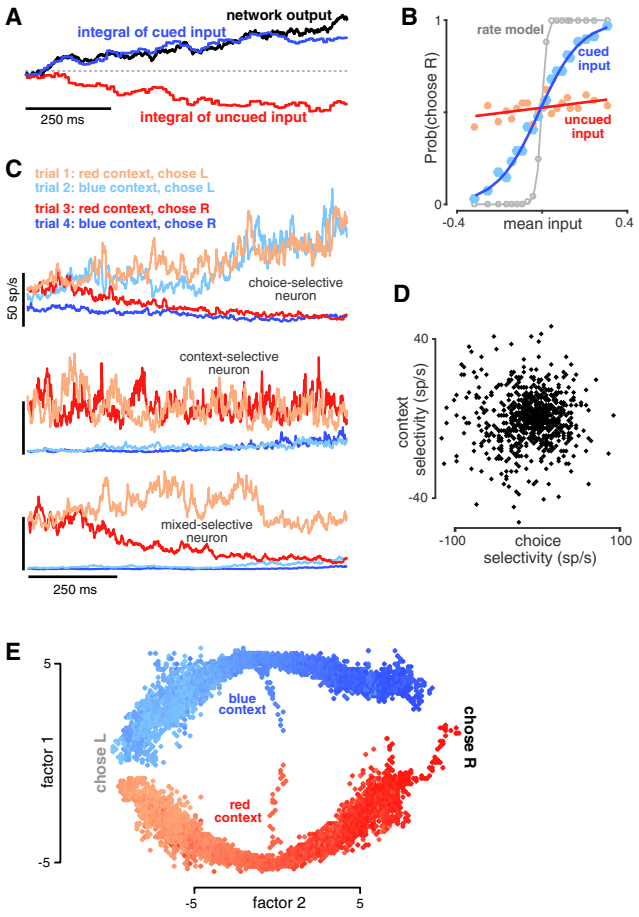

Figure 7. A contextual integration task

(A) Network output (black) for a single example trial. The output correctly tracks the true integral of the cued sensory input (blue) and ignores the uncued sensory input, whose integral is shown in red.

(B) Probability of the spiking network making a rightward choice as a function of the mean input value of the cued (blue) and uncued (red) sensory inputs. The network successfully renders its decision based on the cued input with only modest influence of the uncued input. However, due to the sensitivity of integration to small amounts of noise injected into the factors due to spiking variability, the spiking network is less sensitive to the mean of the cued input than the rate model upon which it is based (gray). Each point is computed across many trials with similar mean input values. Best-fit curves computed by logistic regression.

(C) The factor-based rate, $r_n^F(t)$, for three example neurons and four trials. The three examples include a choice-selective neuron (top), a context-selective neuron (middle), and a mixed-selective neuron (bottom). The choice-selective neuron responds strongly during left-choice trials ("L") and weakly to right-choice trials ("R") in both contexts. The context-selective neuron responds strongly in the red context for both choices. The mixed-selective neuron only responds strongly during red-context left-choice trials.

(D) Scatterplot, with one point per neuron, illustrating that choice selectivity (horizontal axis) and context selectivity (vertical axis) were typically mixed and unrelated. Choice selectivity was defined as the trial-averaged difference in firing rate between left-choice and right-choice trials. Context selectivity was defined as the trial-averaged difference in firing rate between blue-context and red-context trials.

(E) Two-dimensional state-space portrait for the first two factors generated by the spiking network. Each point is the average value across many trials and timepoints that were from the same context and shared similar values of factor two. Color (red versus blue) indicates the cued context. Shade intensity indicates the average value (for that set of trials and timepoints) of the true integral of the cued sensory input. Darker shades indicate more positive values of the true integral (and thus a greater probability that right should be chosen in the future) and lighter shades indicate more negative values. In this network, factor 2 happened to align well with the "decision variable" used by the network. Thus, positive values of factor 2 are associated with more positive integrals/darker shading.

Two-task network

For our spiking networks, P dimensions captured factor-related activity, by design. The remaining $N - P$ dimensions captured signals that were effectively noise. Thus, only a minority of neural dimensions contain task-relevant signals. This observation relates to $\boldsymbol{J}$ having a low-rank task-relevant component and a high-rank task-irrelevant component. For simplicity, we modeled the task-irrelevant component as random. An intriguing possibility is that, in biological spiking networks, components of connectivity that do not contribute to the present task (and appear to generate noise) do contribute to other tasks. This would be consistent with the finding that different computations often use different dimensions, both within task[44,74] and across effectors.[50,82]

To explore this possibility, we trained a spiking network to perform both the cycling and reaching tasks (Figure 8A). We encouraged the network to use task-specific factors (see STAR Methods). Training was successful: neural dimensions that were computationally essential in one task—i.e., reflected factors whose dynamics produced deltoid activity—were unused in the other task (Figure 8B). Network performance was comparable to that of single-task networks (median error of 3%). When a factor was not used (i.e., the corresponding task was not being performed), activity in that dimension was similar to activity in the "non-factor dimensions" that captured small "noisy" fluctuations (Figure S4). Thus, while a given computation may be performed by a low-dimensional set of factors, this does not necessarily imply a fixed set of computationally relevant dimensions.

Separation of computations by dimensions did not create anatomical separation; most neurons' firing rates were modulated during both tasks. Furthermore, as in all networks we trained, each synapse conveyed both computationally relevant signals and non-factor-based signals during both tasks. This cautions against the hope that factor-based and non-factor-based components might be inferred from anatomy alone. One might have hoped to leverage the assumption that $\boldsymbol{J}_{\text{fac}}$ is low rank, but this idealization no longer holds for realistic connectivity. Furthermore, aspects of $\boldsymbol{J}_{\text{fac}}$ connectivity critical to one task may not produce meaningful factors in another task (and are thus more appropriately considered part of $\boldsymbol{J}_0$). Although our approach decomposes connectivity into two components, the key separation is not anatomical but physiological: for a given task, every neuron receives both a factor-based and a non-factor-based input, yielding spiking that appears to probabilistically reflect a factor-based firing rate.

DISCUSSION

Our goal was to establish whether and how network function relates to concepts employed by experimental neuroscientists

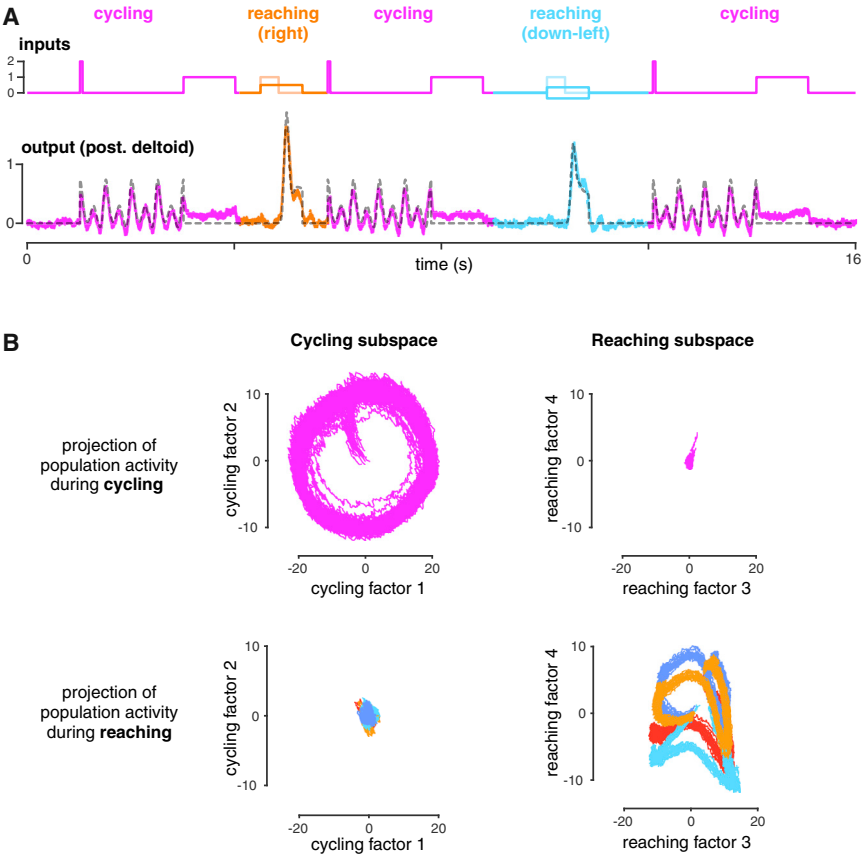

Figure 8. A network that performs two tasks

(A) Network inputs (top) and network output (bottom) of the trained model. Data for reaching trials are colored as in Figure 6, and data for cycling trials are colored in magenta. Inputs for the reaching task were as in Figure 6: a two-dimensional direction signal and a "go" cue. For the cycling task, a pulse prompted cycling (as it did for the network in Figure 2). We also included a "stop" pulse to instruct the network to stop cycling before the start of the next trial. The spiking network was only required to produce a single EMG (the posterior deltoid) as an output because this was the only common EMG recording across both tasks. Neurons received inputs for the different tasks via different input weights, but they are plotted on the same axis for visualization.

(B) Factor trajectories for both tasks in a "cycling subspace" (the first two cycling factors) and a "reaching subspace" (the third and fourth reaching factors, which were more strongly direction-dependent than the first two reaching factors). Trajectories show 20 trials of cycling (magenta) and 20 trials for each of four reach directions (blue, orange, red, and cyan). For visual clarity, only three trials are plotted for each reach direction in the bottom left.

when analyzing population responses. We found that the concept of factor was helpful in understanding network computation and enabled a robust and flexible training method. Using factors as targets allowed us to train spiking networks to perform a wide variety of tasks, which would have been challenging or impossible using other methods. Importantly, factors were chosen to be appropriate "internal" signals for performing the target computation. Such care was critical—in most cases we studied, networks with realistic firing rates and spiking variability could not robustly perform the task unless training included a "complete" set of target factors. The utility of factors as training targets speaks to their centrality; for this network class, computation is best understood at the factor level. Tracing spiking-level causality is extremely challenging, especially as spiking differs greatly even when computation repeats. In contrast, causality is readily summarized at the factor level. These findings support the growing assumption that factors provide a useful way of describing data and linking to models.

Our results demonstrate that spiking networks can compute in a regime where factors provide a valid and useful abstraction describing computational mechanism. Yet, there exist multiple regimes in which spiking networks can operate; not all will have properties adequately summarized at the factor level. The utility of the factor-based abstraction seems most secure for brain areas that exhibit the empirical properties that motivated our modeling choices: mixed selectivity, firing rate covariation among neurons, and asynchronous spiking. These properties are common in cortex, especially anterior to the central sulcus. We also stress that there may be cases where factor-based dynamics are influenced by phenomena best described by other conceptual frameworks (e.g., spike synchrony or oscillations). Further study is needed to ascertain when factors provide a useful abstraction, and when that abstraction must be supplemented by other levels of description. A related point is that state-space trajectories do not, on their own, constitute explanations. They provide explanatory power only when they test or motivate mechanistic hypotheses regarding how factor-level dynamics perform computations.

Mechanistic underpinnings of commonly used concepts

Spiking variability reflected high-dimensional, likely chaotic dynamics,[52,53] whereas factor-level dynamics were lower dimensional and reliable. The presence of two synaptic-input components—one computationally meaningful and one essentially noise—meshes with the standard assumption that neurons exhibit both shared rate variability and private spiking noise[56,80,83,84] and justifies the many analysis methods that assume such a division, including latent-factor models (e.g., Pandarinath et al.[11]) and coupled generalized linear models (GLMs) (e.g., Yates et al.[58]). Although statistically appropriate, the private versus shared distinction is mechanistically incorrect in our networks; both components derive from the population. This does not imply abandoning standard statistical assumptions such as modeling spikes as Poisson samples following an underlying rate; such statistical models remain useful and networks of such Poisson spiking neurons could be trained using our approach. Another

subtlety to be explored is that the distinction between dimensions containing reliable versus unreliable signals may be more gradual than considered here. This is true both because reliable dimensions may be task-dependent (as in our two-task network), and because there may exist weak factors that are marginally reliable.

Firing rate is a widely accepted abstraction of spiking activity, despite its unclear biophysical basis. In contrast, factors are sometimes viewed with suspicion as overly abstract. Our networks argue that the definition of firing rate should be based on factors and that factors should be considered "primary." Although mechanistically defined factor-based rates behave similarly to statistically inferred rates, the mechanistic definition demands reconsidering some intuitions. Consider the recent controversy regarding whether, during decision-making, single-neuron rates are ramp-like or step-like.[85,86] From the traditional perspective, a natural question is whether, on a single trial, all rates step together or whether each neuron's rate steps at a unique time. Given the factor-based definition of rate, the latter possibility is unlikely or even poorly defined. Because a neuron's rate is based on shared factors, it is unlikely to step at a unique time. One might propose many factors, each stepping at different moments. Yet, because a neuron's rate is typically determined by many factors, this would result in either ramping (if all weights were positive) or multiphasic responses, not steps at unique times. Thus, a possibility that seemed likely from a traditional perspective is much less natural given a factor-based definition of rate.

Our definition of the firing rate also clarifies that rates can change rapidly (as rapidly as factors). Rate coding may appear inextricably linked with longer timescales, as opposed to temporal codes that use fine timescales.[24,87] Yet, if a firing rate describes a "functional group" of identically responsive but asynchronously active neurons, rate can change quickly.[28] Our definition generalizes this view to situations that lack clusters of similarly tuned neurons.[9,14,29,45]

Our proposed definition of firing rate speaks to emerging methods that infer single-trial firing rates. The power of bespoke methods like LFADS (and others cited above) originates from assumptions they employ when data are limited. For example, LFADS assumes consistent internal dynamics, providing a reasonable constraint in data-limited cases. However, using complex operations to infer rates does not imply that the definition of firing rate, or factors, is complex. On the contrary, factors are simply weighted sums of spiking-neuron activity. Firing rates are weighted sums of factors passed through a rectifying nonlinearity.

Different approaches to constructing spiking networks

Seminal work established that spiking networks can emulate continuous dynamical systems.[23,30,33] Our approach achieves a similar end, but the explicit focus on factors both establishes them as mechanistically central and expands the space of emulatable dynamics by allowing target-factors to be derived from rate models. Computation can be understood at the level of the factors, which influence their own future values via a population of spiking neurons. Importantly, this occurs despite realistically variable spiking, and without the need for special mechanisms to correct spiking variations. The factor-level focus, and the use of RLS, should make it possible to train spiking networks on nearly any task that can presently be performed by rate-based networks, greatly extending the set of available network training tools.

Using factors as training targets contrasts with typical training approaches that focus on minimizing output error. In output-focused methods, training aims to indirectly produce internal activity patterns appropriate to perform the underlying computations. This approach is powerful in rate networks but much less so in spiking networks, which often fail to learn the necessary internal signals. Nicola and Clopath[39] showed that simple patterns and sequences could be learned via an output-focused approach (FORCE) in a spiking network, although certain revealing errors were common. They thus deployed a helpful trick: training additional temporally structured outputs. Other approaches took this insight even further, by using internal training targets for every network unit.[37,49,88] Our factor-based approach can be seen as a useful "middle ground" strategy. If computationally appropriate factors can be identified and used as training targets, there is no need to hope that additional factors will emerge (we show that they mostly do not) and no need to include additional, redundant training targets. This approach increases noise robustness[49] in the face of realistic spiking variability and thus expands the range of trainable tasks.

Backpropagation-based learning, which also seeks to exclusively minimize output error, has been applied to spiking models.[36,41,42] Yet, how to achieve general success with this approach remains unclear. We were unable to use one such approach that has shown promise, surrogate-gradient methods,[41,42] to train spiking networks to perform the tasks in the present study. We suspect these methods could be made to work with a carefully designed objective function. If so, a likely consequence is that they will identify similar solutions to those that emerge in our procedure of instantiating rate-network-derived factors.

An alternative approach, which has been quite successful in spiking networks, is to engineer them to emulate dynamical systems.[23,30–32,34,35,40,89] For such approaches, state variables are effectively factors, and the importance of embedding key computations within low-rank connectivity has been noted.[23,32,46,48] Dynamics must typically be specified as a closed-form dynamical system, although ad hoc approaches have been successful when this is not possible.[31] Our method is related but can be readily used on any appropriate factors and highlights the vast potential of trained rate models for deriving factors.

Although achieving both stable dynamics and considerable spiking variability was not a goal of early approaches,[23] it was critical to our goal of building models that guide interpretation of empirical data. Achieving this goal was greatly aided by the RLS approach to stabilizing factor trajectories. An alternative approach to achieving robust dynamics alongside considerable spiking variability[31,32,40] makes the strong assumption of fine timescale "corrective" dynamics, which both gives rise to spiking variability and manages its impact on network performance. Our approach demonstrates that, even without such a mechanism, stable factor-level dynamics can co-exist with realistic spiking variability. Furthermore, our results suggest that spiking variability may be a consequence not of corrective dynamics, but of capacity constraints: high-rank connectivity, despite increasing spiking

variability, may be necessary to realize multiple computations within the same network (e.g., Logiaco et al.[90]).

Historically, firing rate models have often been considered approximations of spiking models, with individual rate units representing individual spiking neurons.[91–94] It is possible to build rate and spiking models with neuron-to-neuron correspondence,[38] although whether this approach can yield noise robustness is presently unclear. Our approach is related but achieves robustness to spiking variability by using RLS to train factors, without attempting to achieve a rate-to-spiking-neuron correspondence. An alternative approach to linking rate and spiking models is to model each rate unit as a pool of spiking neurons.[26,27] This can be convenient and effective, but there are reasons to prefer factor-level correspondence. Neurons in many areas show great tuning heterogeneity, inconsistent with pools of similarly tuned neurons. Furthermore, from a purely model-building perspective, a factor-level correspondence is more efficient. Because rate-network units are correlated, a rate-to-spiking-pool correspondence repeatedly constructs correlated signals. A factor-level correspondence avoids this redundancy by identifying correlated activity patterns (i.e., rate-network factors) and using them as targets, greatly reducing the necessary number of spiking neurons. This approach embraces the view that, even when networks use very different types of units, they can still perform the same computation using the same factors, making the factor level a natural point of comparison.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - Spiking network model
  - Balancing task performance and spiking irregularity
  - Dividing the synaptic input into factor-based and non-factor-based components
  - Deriving target factors
  - Quantifying performance, factor-based input variability, and spiking irregularity in trained spiking models
  - Computing the flow field of the factors
  - Computing $r_n^F(t)$
  - LFADS analysis
  - Tangling analysis
  - Learning sparsely connected and Dale's Law obeying networks
  - Task details

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.neuron.2022.12.007.

ACKNOWLEDGMENTS

We thank Antonio Lara and Abigail Russo of the Churchland lab for providing the reaching EMG data and the cycling EMG and factor data. We thank members of the Churchland lab, Jonathan Pillow and his lab, and Carlos Brody and his lab for comments and feedback on this manuscript. We thank Stefano Fusi and Misha Tsodyks for feedback on the work as it developed. Research supported by NIH grant MH093338, the Simons Collaboration for the Global Brain, the Grossman Charitable Trust, the Gatsby Charitable Foundation, the Swartz Foundation, the McKnight Foundation, NIH 1U19NS104649, and NSF NeuroNex award DBI-1707398. B.D. was supported by a National Science Foundation Graduate Research Fellowship while performing this study.

AUTHOR CONTRIBUTIONS

B.D., L.F.A., and M.M.C. conceived and designed the conceptual and technical aspects of the study. B.D. constructed the primary network models, analyzed the simulated data, and generated visualizations of the simulated data. D.S. constructed supporting network models, provided software, and provided conceptual expertise during development of the study. B.D., L.F.A., and M.M.C. wrote the text. All authors edited and reviewed the text. L.F.A. and M.M.C. supervised and funded the study.

DECLARATION OF INTERESTS

L.F.A. serves on the advisory board of *Neuron*. D.S. is employed as a research scientist by Meta (Meta Reality Labs); his work there is unrelated to this study.

Received: December 21, 2020
Revised: June 17, 2022
Accepted: December 5, 2022
Published: January 10, 2023

REFERENCES

1. Gold, J.I., and Shadlen, M.N. (July 2007). The neural basis of decision making. Annu. Rev. Neurosci. *30*, 535–574. https://doi.org/10.1146/annurev.neuro.29.051605.113038.
2. Hanks, T.D., Kopec, C.D., Brunton, B.W., Duan, C.A., Erlich, J.C., and Brody, C.D. (January 2015). Distinct relationships of parietal and prefrontal cortices to evidence accumulation. Nature *520*, 220–223. https://doi.org/10.1038/nature14066.
3. Aoi, M.C., Mante, V., and Pillow, J.W. (October 2020). Prefrontal cortex exhibits multidimensional dynamic encoding during decision-making. Nat. Neurosci. *23*, 1410–1420. https://doi.org/10.1038/s41593-020-0696-5.
4. Barack, D.L., and Krakauer, J.W. (April 2021). Two views on the cognitive brain. Nat. Rev. Neurosci. *22*, 359–371. https://doi.org/10.1038/s41583-021-00448-6.
5. Briggman, K.L., Abarbanel, H.D.I., and Kristan, W.B. (February 2005). Optical imaging of neuronal populations during decision-making. Science *307*, 896–901. https://doi.org/10.1126/science.1103736.
6. Bruno, A.M., Frost, W.N., and Humphries, M.D. (2017). A spiral attractor network drives rhythmic locomotion. eLife *6*, e27342. https://doi.org/10.7554/eLife.27342.
7. Duncker, L., and Sahani, M. (October 2021). Dynamics on the manifold: identifying computational dynamical activity from neural population recordings. Curr. Opin. Neurobiol. *70*, 163–170. https://doi.org/10.1016/j.conb.2021.10.014.
8. Huang, C., Ruff, D.A., Pyle, R., Rosenbaum, R., Cohen, M.R., and Doiron, B. (January 2019). Circuit models of low-dimensional shared variability in cortical networks. Neuron *101*, 337–348.e4. https://doi.org/10.1016/j.neuron.2018.11.034.
9. Mante, V., Sussillo, D., Shenoy, K.V., and Newsome, W.T. (November 2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature *503*, 78–84. https://doi.org/10.1038/nature12742.

10. Mazzucato, L., Fontanini, A., and La Camera, G. (2016). Stimuli reduce the dimensionality of cortical activity. Front. Syst. Neurosci. *10*, 11. https://doi.org/10.3389/fnsys.2016.00011.

11. Pandarinath, C., O'Shea, D.J., Collins, J., Jozefowicz, R., Stavisky, S.D., Kao, J.C., Trautmann, E.M., Kaufman, M.T., Ryu, S.I., Hochberg, L.R., et al. (September 2018). Inferring single-trial neural population dynamics using sequential auto-encoders. Nat. Methods *15*, 805–815. https://doi.org/10.1038/s41592-018-0109-9.

12. Recanatesi, S., Ocker, G.K., Buice, M.A., and Shea-Brown, E. (July 2019). Dimensionality in recurrent spiking networks: global trends in activity and local origins in connectivity. PLOS Comp. Biol. *15*, e1006446. https://doi.org/10.1371/journal.pcbi.1006446.

13. Remington, E.D., Egger, S.W., Narain, D., Wang, J., and Jazayeri, M. (October 2018). A dynamical systems perspective on flexible motor timing. Trends Cogn. Sci. *22*, 938–952. https://doi.org/10.1016/j.tics.2018.07.010.

14. Russo, A.A., Bittner, S.R., Perkins, S.M., Seely, J.S., London, B.M., Lara, A.H., Miri, A., Marshall, N.J., Kohn, A., Jessell, T.M., et al. (February 2018). Motor cortex embeds muscle-like commands in an untangled population response. Neuron *97*, 953–966.e8. https://doi.org/10.1016/j.neuron.2018.01.004.

15. Sohn, H., Narain, D., Meirhaeghe, N., and Jazayeri, M. (September 2019). Bayesian computation through cortical latent dynamics. Neuron *103*, 934–947.e5. https://doi.org/10.1016/j.neuron.2019.06.012.

16. Stopfer, M., Jayaraman, V., and Laurent, G. (September 2003). Intensity versus identity coding in an olfactory system. Neuron *39*, 991–1004. https://doi.org/10.1016/j.neuron.2003.08.011.

17. Wei, Z., Inagaki, H., Li, N., Svoboda, K., and Druckmann, Shaul (January 2019). An orderly single-trial organization of population dynamics in premotor cortex predicts behavioral variability. Nat. Commun. *10*, 216. https://doi.org/10.1038/s41467-018-08141-6.

18. Williamson, R.C., Cowley, B.R., Litwin-Kumar, A., Doiron, B., Kohn, A., Smith, M.A., and Yu, B.M. (December 2016). Scaling properties of dimensionality reduction for neural populations and network models. PLoS Comp. Biol. *12*, e1005141. https://doi.org/10.1371/journal.pcbi.1005141.

19. Williamson, R.C., Doiron, B., Smith, M.A., and Yu, B.M. (April 2019). Bridging large-scale neuronal recordings and large-scale network models using dimensionality reduction. Curr. Opin. Neurobiol. *55*, 40–47. https://doi.org/10.1016/j.conb.2018.12.009.

20. Churchland, M.M., Cunningham, J.P., Kaufman, M.T., Foster, J.D., Nuyujukian, P., Ryu, S.I., and Shenoy, K.V. (June 2012). Neural population dynamics during reaching. Nature *487*, 51–56. https://doi.org/10.1038/nature11129.

21. Sussillo, D., Churchland, M.M., Kaufman, M.T., and Shenoy, K.V. (June 2015). A neural network that finds a naturalistic solution for the production of muscle activity. Nat. Neurosci. *18*, 1025–1033. https://doi.org/10.1038/nn.4042.

22. Barak, O., and Romani, S. (March 2021). Mapping low-dimensional dynamics to high-dimensional neural activity: A derivation of the ring model from the neural engineering framework. Neural Comput. *33*, 827–852. https://doi.org/10.1162/neco_a_01361.

23. Eliasmith, C. (June 2005). A unified approach to building and controlling spiking attractor networks. Neural Comput. *17*, 1276–1314. https://doi.org/10.1162/0899766053630332.

24. Bialek, W., Rieke, F., de Ruyter van Steveninck, R.R., and Warland, D. (June 1991). Reading a neural code. Science *252*, 1854–1857. https://doi.org/10.1126/science.2063199.

25. Brette, R. (November 2015). Philosophy of the spike: rate-based vs. spike-based theories of the brain. Front. Syst. Neurosci. *9*, 151. https://doi.org/10.3389/fnsys.2015.00151.

26. Hennequin, G., Vogels, T.P., and Gerstner, W. (June 2014). Optimal control of transient dynamics in balanced networks supports generation of complex movements. Neuron *82*, 1394–1406. https://doi.org/10.1016/j.neuron.2014.04.045.

27. Litwin-Kumar, A., and Doiron, B. (September 2012). Slow dynamics and high variability in balanced cortical networks with clustered connections. Nat. Neurosci. *15*, 1498–1505. https://doi.org/10.1038/nn.3220.

28. Shadlen, M.N., and Newsome, W.T. (May 1998). The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. J. Neurosci. *18*, 3870–3896. https://doi.org/10.1523/JNEUROSCI.18-10-03870.1998.

29. Churchland, M.M., and Shenoy, K.V. (June 2007). Temporal complexity and heterogeneity of single-neuron activity in premotor and motor cortex. J. Neurophysiol. *97*, 4235–4257. https://doi.org/10.1152/jn.00095.2007.

30. Abbott, L.F., DePasquale, B., and Memmesheimer, R.-M. (February 2016). Building functional networks of spiking model neurons. Nat. Neurosci. *19*, 350–355. https://doi.org/10.1038/nn.4241.

31. Alemi, A., Machens, C., Deneve, S., and Slotine, J.-J. (April 2018). Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules. AAAI *32*. https://doi.org/10.1609/aaai.v32i1.11320.

32. Boerlin, M., Machens, C.K., and Denève, S. (November 2013). Predictive coding of dynamical variables in balanced spiking networks. PLoS Comp. Biol. *9*, e1003258. https://doi.org/10.1371/journal.pcbi.1003258.

33. Denève, S., and Machens, C.K. (February 2016). Efficient codes and balanced networks. Nat. Neurosci. *19*, 375–382. https://doi.org/10.1038/nn.4243.

34. DePasquale, B., Churchland, M.M., and Abbott, L.F. (2016). Using firing-rate dynamics to train recurrent networks of spiking model neurons. https://doi.org/10.48550/ARXIV.1601.07620.

35. Gilra, A., and Gerstner, W. (November 2017). Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network. eLife *6*, e28295. https://doi.org/10.7554/eLife.28295.

36. Huh, D., and Sejnowski, T.J. (2018). Gradient descent for spiking neural networks. In Advances in Neural Information Processing Systems, *31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds. (Curran Associates, Inc.). https://proceedings.neurips.cc/paper/2018/file/185e65bc40581880c4f2c82958de8cfe-Paper.pdf.

37. Kim, C.M., and Chow, C.C. (2018). Learning recurrent dynamics in spiking networks. eLife *7*, e37124. https://doi.org/10.7554/eLife.37124.

38. Kim, R., Li, Y., and Sejnowski, T.J. (October 2019). Simple framework for constructing functional spiking recurrent neural networks. Proc. Natl. Acad. Sci. USA *116*, 22811–22820. https://doi.org/10.1073/pnas.1905926116.

39. Nicola, W., and Clopath, C. (December 2017). Supervised learning in spiking neural networks with FORCE training. Nat. Commun. *8*, 2208. https://doi.org/10.1038/s41467-017-01827-3.

40. Thalmeier, D., Uhlmann, M., Kappen, H.J., and Memmesheimer, R.-M. (June 2016). Learning universal computations with spikes. PLoS Comp. Biol. *12*, e1004895. https://doi.org/10.1371/journal.pcbi.1004895.

41. Zenke, F., and Ganguli, S. (June 2018). SuperSpike: supervised learning in multilayer spiking neural networks. Neural Comput. *30*, 1514–1541. https://doi.org/10.1162/neco_a_01086.

42. Zenke, F., and Vogels, T.P. (2021). The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. Neural Comput. *33*, 899–925. https://doi.org/10.1162/neco_a_01367.

43. Trautmann, E.M., Stavisky, S.D., Lahiri, S., Ames, K.C., Kaufman, M.T., O'Shea, D.J., Vyas, S., Sun, Xulu, Ryu, S.I., Ganguli, S., et al. (July 2019). Accurate estimation of neural population dynamics without spike sorting. Neuron *103*, 292–308.e4. https://doi.org/10.1016/j.neuron.2019.05.003.

44. Machens, C.K., Romo, R., and Brody, C.D. (January 2010). Functional, but not anatomical, separation of "what" and "when" in prefrontal cortex. J. Neurosci. *30*, 350–360. https://doi.org/10.1523/JNEUROSCI.3276-09.2010.

45. Rigotti, M., Barak, O., Warden, M.R., Wang, X.-J., Daw, N.D., Miller, E.K., and Fusi, S. (May 2013). The importance of mixed selectivity in complex cognitive tasks. Nature *497*, 585–590. https://doi.org/10.1038/nature12160.

46. Mastrogiuseppe, F., and Ostojic, S. (August 2018). Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. Neuron *99*, 609–623.e29. https://doi.org/10.1016/j.neuron.2018.07.003.

47. Schuessler, F., Mastrogiuseppe, F., Dubreuil, A., Ostojic, S., and Barak, O. (2020). The interplay between randomness and structure during learning in RNNs. Adv. Neural Inf. Process. Syst. *33*, 13352–13362. https://proceedings.neurips.cc/paper/2020/file/9ac1382fd8fc4b631594aa135d16ad75-Paper.pdf.

48. Sussillo, D., and Abbott, L.F. (August 2009). Generating coherent patterns of activity from chaotic neural networks. Neuron *63*, 544–557. https://doi.org/10.1016/j.neuron.2009.07.018.

49. DePasquale, B., Cueva, C.J., Rajan, K., Escola, G.S., and Abbott, L.F. (February 2018). full-FORCE: A target-based method for training recurrent networks. PLoS One *13*, e0191527. https://doi.org/10.1371/journal.pone.0191527.

50. Ames, K.C., and Churchland, M.M. (2019). Motor cortex signals for each arm are mixed across hemispheres and neurons yet partitioned within the population response. eLife *8*, e46159. https://doi.org/10.7554/eLife.46159.

51. Churchland, M.M., Yu, B.M., Ryu, S.I., Santhanam, G., and Shenoy, K.V. (2006). Neural variability in premotor cortex provides a signature of motor preparation. J. Neurosci. *26*, 3697–3712. https://doi.org/10.1523/JNEUROSCI.3762-05.2006.

52. van Vreeswijk, C., and Sompolinsky, H. (August 1998). Chaotic balanced state in a model of cortical circuits. Neural Comput. *10*, 1321–1371. https://doi.org/10.1162/089976698300017214.

53. Zillmer, R., Brunel, N., and Hansel, D. (March 2009). Very long transients, irregular firing, and chaotic dynamics in networks of randomly connected inhibitory integrate-and-fire neurons. Phys. Rev. E Stat. Nonlin. Soft Matter Phys. *79*, 031909. https://doi.org/10.1103/PhysRevE.79.031909.

54. Churchland, M.M., and Abbott, L.F. (October 2012). Two layers of neural variability. Nat. Neurosci. *15*, 1472–1474. https://doi.org/10.1038/nn.3247.

55. Park, I.M., Meister, M.L.R., Huk, A.C., and Pillow, J.W. (August 2014). Encoding and decoding in parietal cortex during sensorimotor decision-making. Nat. Neurosci. *17*, 1395–1403. https://doi.org/10.1038/nn.3800.

56. Churchland, A.K., Kiani, R., Chaudhuri, R., Wang, X.J., Pouget, A., and Shadlen, M.N. (February 2011). Variance as a signature of neural computations during decision making. Neuron *69*, 818–831. https://doi.org/10.1016/j.neuron.2010.12.037.

57. Yu, B.M., Cunningham, J.P., Santhanam, G., Ryu, S.I., Shenoy, K.V., and Sahani, M. (July 2009). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. J. Neurophysiol. *102*, 614–635. https://doi.org/10.1152/jn.90941.2008.

58. Yates, J.L., Park, I.M., Katz, L.N., Pillow, J.W., and Huk, A.C. (July 2017). Functional dissection of signal and noise in MT and LIP during decision-making. Nat. Neurosci. *20*, 1285–1292. https://doi.org/10.1038/nn.4611.

59. Gao, Y., Archer, E., Paninski, L., and Cunningham, J.P. (2016). Linear dynamical neural population models through nonlinear embeddings. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS, pp. 163–171. http://www.stat.columbia.edu/~cunningham/pdf/GaoNIPS2016.pdf.

60. Hernandez, D., Moretti, A.K., Wei, Z., Saxena, S., Cunningham, J., and Paninski, L. (2020). Nonlinear evolution via spatially-dependent linear dynamics for electrophysiology and calcium data. https://doi.org/10.48550/ARXIV.1811.02459.

61. Macke, J.H., Buesing, L., Cunningham, J.P., Byron, M.Y., Shenoy, K.V., and Sahani, M. (2011). Empirical models of spiking in neural populations. In Advances in Neural Information Processing Systems, *24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, eds. (Curran Associates, Inc.). https://proceedings.neurips.cc/paper/2011/file/7143d7fbadfa4693b9eec507d9d37443-Paper.pdf.

62. Saxena, S., and Cunningham, J.P. (April 2019). Towards the neural population doctrine. Curr. Opin. Neurobiol. *55*, 103–111. https://doi.org/10.1016/j.conb.2019.02.002.

63. Wu, A., Roy, N.A., Keeley, S., and Pillow, J.W. (2017). Gaussian process based nonlinear latent structure discovery in multivariate spike train data. In Advances in Neural Information Processing Systems, *30*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds. (Curran Associates, Inc.). https://proceedings.neurips.cc/paper/2017/file/b3b4d2dbedc99fe843fd3dedb02f086f-Paper.pdf.

64. Zhao, Y., and Park, I.M. (May 2017). Variational latent gaussian process for recovering single-trial dynamics from population spike trains. Neural Comput. *29*, 1293–1316. https://doi.org/10.1162/NECO_a_00953.

65. Cunningham, J.P., and Yu, B.M. (August 2014). Dimensionality reduction for large-scale neural recordings. Nat. Neurosci. *17*, 1500–1509. https://doi.org/10.1038/nn.3776.

66. Shenoy, K.V., Sahani, M., and Churchland, M.M. (July 2013). Cortical control of arm movements: A dynamical systems perspective. Annu. Rev. Neurosci. *36*, 337–359. https://doi.org/10.1146/annurev-neuro-062111-150509.

67. Herbert, E., and Ostojic, S. (August 2022). The impact of sparsity in low-rank recurrent neural networks. PLoS Comp. Biol. *18*, e1010426. https://doi.org/10.1371/journal.pcbi.1010426.

68. Jaeger, H., and Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science *304*, 78–80. https://doi.org/10.1126/science.1091277.

69. Barak, O. (2017). Recurrent neural networks as versatile tools of neuroscience research. Curr. Opin. Neurobiol. *46*, 1–6. https://doi.org/10.1016/j.conb.2017.06.003.

70. Pollock, E., and Jazayeri, M. (August 2020). Engineering recurrent neural networks from task-relevant manifolds and dynamics. PLoS Comp. Biol. *16*, e1008128. https://doi.org/10.1371/journal.pcbi.1008128.

71. Sussillo, D. (April 2014). Neural circuits as computational dynamical systems. Curr. Opin. Neurobiol. *25*, 156–163. https://doi.org/10.1016/j.conb.2014.01.008.

72. Yang, G.R., Joglekar, M.R., Song, H.F., Newsome, W.T., and Wang, X.-J. (January 2019). Task representations in neural networks trained to perform many cognitive tasks. Nat. Neurosci. *22*, 297–306. https://doi.org/10.1038/s41593-018-0310-2.

73. Michaels, J.A., Dann, B., and Scherberger, H. (November 2016). Neural population dynamics during reaching are better explained by a dynamical system than representational tuning. PLOS Comp. Biol. *12*, e1005175. https://doi.org/10.1371/journal.pcbi.1005175.

74. Elsayed, G.F., Lara, A.H., Kaufman, M.T., Churchland, M.M., and Cunningham, J.P. (October 2016). Reorganization between preparatory and movement population responses in motor cortex. Nat. Commun. *7*, 13239. https://doi.org/10.1038/ncomms13239.

75. Lara, A.H., Cunningham, J.P., and Churchland, M.M. (July 2018). Different population dynamics in the supplementary motor area and motor cortex during reaching. Nat. Commun. *9*, 2754. https://doi.org/10.1038/s41467-018-05146-z.

76. Georgopoulos, A.P., Kalaska, J.F., Caminiti, R., and Massey, J.T. (November 1982). On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. J. Neurosci. *2*, 1527–1537. https://doi.org/10.1523/JNEUROSCI.02-11-01527.1982.

77. Churchland, M.M., Cunningham, J.P., Kaufman, M.T., Ryu, S.I., and Shenoy, K.V. (November 2010). Cortical preparatory activity: representation of movement or first cog in a dynamical machine? Neuron *68*, 387–400. https://doi.org/10.1016/j.neuron.2010.09.015.

78. Kaufman, M.T., Churchland, M.M., Santhanam, G., Yu, B.M., Afshar, A., Ryu, S.I., and Shenoy, K.V. (August 2010). Roles of monkey premotor neuron classes in movement preparation and execution. J. Neurophysiol. *104*, 799–810. https://doi.org/10.1152/jn.00231.2009.

79. Churchland, M.M., Afshar, A., and Shenoy, K.V. (December 2006). A central source of movement variability. Neuron *52*, 1085–1096. https://doi.org/10.1016/j.neuron.2006.10.034.

80. Churchland, M.M., Yu, B.M., Cunningham, J.P., Sugrue, L.P., Cohen, M.R., Corrado, G.S., Newsome, W.T., Clark, A.M., Hosseini, P., Scott, B.B., et al. (February 2010). Stimulus onset quenches neural variability: a widespread cortical phenomenon. Nat. Neurosci. *13*, 369–378. https://doi.org/10.1038/nn.2501.

81. Werbos, P.J. (1990). Backpropagation through time: what it does and how to do it. Proc. IEEE *78*, 1550–1560. https://doi.org/10.1109/5.58337.

82. Heming, E.A., Cross, K.P., Takei, T., Cook, D.J., and Scott, S.H. (2019). Independent representations of ipsilateral and contralateral limbs in primary motor cortex. eLife *8*, e48190. https://doi.org/10.7554/eLife.48190.

83. Golub, M.D., Sadtler, P.T., Oby, E.R., Quick, K.M., Ryu, S.I., Tyler-Kabara, E.C., Batista, A.P., Chase, S.M., Yu, B.M., and Byron, M. (March 2018). Yu. Learning by neural reassociation. Nat. Neurosci. *21*, 607–616. https://doi.org/10.1038/s41593-018-0095-3.

84. Sadtler, P.T., Quick, K.M., Golub, M.D., Chase, S.M., Ryu, S.I., Tyler-Kabara, E.C., Yu, B.M., and Batista, A.P. (August 2014). Neural constraints on learning. Nature *512*, 423–426. https://doi.org/10.1038/nature13665.

85. Latimer, K.W., Yates, J.L., Meister, M.L.R., Huk, A.C., and Pillow, J.W. (July 2015). NEURONAL MODELING. Single-trial spike trains in parietal cortex reveal discrete steps during decision-making. Science *349*, 184–187. https://doi.org/10.1126/science.aaa4056.

86. Shadlen, M.N., Kiani, R., Newsome, W.T., Gold, J.I., Wolpert, D.M., Zylberberg, A., Ditterich, J., de Lafuente, V., Yang, T., and Roitman, J. (March 2016). Comment on "single-trial spike trains in parietal cortex reveal discrete steps during decision-making". Science *351*, 1406. https://doi.org/10.1126/science.aad3242.

87. Theunissen, F., and Miller, J.P. (June 1995). Temporal encoding in nervous systems: A rigorous definition. J. Comp. Neurosci. *2*, 149–162. https://doi.org/10.1007/BF00961885.

88. Rajan, K., Harvey, C.D., and Tank, D.W. (April 2016). Recurrent network models of sequence generation and memory. Neuron *90*, 128–142. https://doi.org/10.1016/j.neuron.2016.02.009.

89. Boerlin, M., and Denève, S. (February 2011). Spike-based population coding and working memory. PLoS Comp. Biol. *7*, e1001080. https://doi.org/10.1371/journal.pcbi.1001080.

90. Logiaco, L., Abbott, L.F., and Escola, S. (June 2021). Thalamic control of cortical dynamics in a model of flexible motor sequencing. Cell Rep. *35*, 109090. https://doi.org/10.1016/j.celrep.2021.109090.

91. Ermentrout, B. (July 1994). Reduction of conductance-based models with slow synapses to neural nets. Neural Comput. *6*, 679–695. https://doi.org/10.1162/neco.1994.6.4.679.

92. Gerstner, W. (January 1995). Time structure of the activity in neural network models. Phys. Rev. E Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics *51*, 738–758. https://doi.org/10.1103/physreve.51.738.

93. Ostojic, S., and Brunel, N. (January 2011). From spiking neuron models to linear-nonlinear models. PLoS Comp. Biol. *7*, e1001056. https://doi.org/10.1371/journal.pcbi.1001056.

94. Shriki, O., Hansel, D., and Sompolinsky, H. (August 2003). Rate models for conductance-based cortical neuronal networks. Neural Comput. *15*, 1809–1841. https://doi.org/10.1162/08997660360675053.

95. Wei, Z., Lin, B.-J., Chen, T.-W., Daie, K., Svoboda, K., and Druckmann, Shaul (September 2020). A comparison of neuronal population dynamics measured with calcium imaging and electrophysiology. PLoS Comp. Biol. *16*, e1008198. https://doi.org/10.1371/journal.pcbi.1008198.

96. Ingrosso, A., and Abbott, L.F. (August 2019). Training dynamically balanced excitatory-inhibitory networks. PLoS One *14*, e0220547. https://doi.org/10.1371/journal.pone.0220547.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Processed empirical data used for network training.	Lab of Mark Churchland. Processed data available at https://github.com/briandepasquale/factor-based-spiking-nets/. Unprocessed data was generated during experiments whose results were reported in Lara et al.[75] (https://doi.org/10.1038/s41467-018-05146-z) and Russo et al.[14] (https://doi.org/10.1016/j.neuron.2018.01.004)	Github: https://doi.org/10.5281/zenodo.7302474
Computer simulated data and original computer code for running data-generating network simulations.	Code was written by the authors. Available at https://github.com/briandepasquale/factor-based-spiking-nets/	Github: https://doi.org/10.5281/zenodo.7302474
Software and algorithms		
Matlab 2016a	**Mathworks**	https://www.mathworks.com/products/matlab.html
Python	Python Software Foundation	https://www.python.org/
Custom computer code for data-generating network simulations, network training, and analysis.	Code was written by the authors. Available at https://github.com/briandepasquale/factor-based-spiking-nets/	Github: https://doi.org/10.5281/zenodo.7302474

RESOURCE AVAILABILITY

Lead contact

Further information and requests should be directed to and will be fulfilled by the lead contact, Brian DePasquale (bddepasq@bu.edu).

Materials availability

This study did not generate new unique reagents.

Data and code availability

- Original data used in this study was computer simulated. Some simulations relied on processed empirical data. Processed empirical data, computer simulated data, and original computer code for running data-generating network simulations has been deposited at https://github.com/briandepasquale/factor-based-spiking-nets/ and is publicly available as of the date of publication. DOIs are listed in the key resources table.
- Custom code for network training and analysis has been deposited at https://github.com/briandepasquale/factor-based-spiking-nets/ and is publicly available as of the date of publication. DOIs are listed in the key resources table.
- Any additional information required to regenerate or reanalyze the data reported in this paper is available from the lead contact upon request.

METHOD DETAILS

Spiking network model

Leaky integrate-and-fire model neurons were used. N designates the number of spiking neurons. The membrane potential of all neurons is denoted by the N-component vector $\boldsymbol{v}(t)$. When v_i reaches a threshold value of 0 mV neuron i fires an action potential and is reset to -10 mV.

Each neuron's spikes were filtered with two characteristic timescales, denoted by two N-dimensional vectors $\boldsymbol{s}^{\mathrm{f}}(t)$ and $\boldsymbol{s}^{\mathrm{s}}(t)$. When neuron i spikes, $s_i^{\mathrm{f}}(t)$ and $s_i^{\mathrm{s}}(t)$ increment by 1. At all other times the presynaptic inputs decay exponentially with a time constant of $\tau_{\mathrm{f}} = 5$ ms and $\tau_{\mathrm{s}} = 100$ ms. For simplicity, we concatenate $s_i^{\mathrm{f}}(t)$ and $s_i^{\mathrm{s}}(t)$ into a $2N$-dimensional vector $\boldsymbol{s}(t)$.

We construct P factors from the filtered spikes. The P-component vector $\boldsymbol{y}(t)$ denotes the network generated factors. A $P\times 2N$ matrix $\boldsymbol{w}$ that is learned (see next subsection) combines the filtered spikes to yield the network generated factors: $\boldsymbol{y}(t) = \boldsymbol{w}\boldsymbol{s}(t)$. Each neuron receives linear combinations of the factors through a $N\times P$ matrix $\boldsymbol{u}$. Because of this, the $N\times 2N$ matrix of recurrent connections that constructs the factors through learning, $\boldsymbol{J}_{\text{fac}}$, is defined as $\boldsymbol{J}_{\text{fac}} = \boldsymbol{u}\boldsymbol{w}$.

To account for synaptic inputs that do not arise from the factors we include a second $N\times 2N$ recurrent connectivity matrix, $\boldsymbol{J}_0$, of unmodified connections. The elements of the first N columns of $\boldsymbol{J}_0$ correspond to the filtered spikes $\boldsymbol{s}^{\text{f}}(t)$ and the elements of the second N columns correspond to $\boldsymbol{s}^{\text{s}}(t)$.

$\boldsymbol{v}(t)$ obeys the following dynamical equation, which is standard for a LIF neuron,

$$\tau_v \frac{d\boldsymbol{v}(t)}{dt} = -\left(\boldsymbol{v}(t) - \boldsymbol{v}_\mu\right) + \boldsymbol{J}_0\boldsymbol{s}(t) + \boldsymbol{J}_{\text{fac}}\boldsymbol{s}(t) + \boldsymbol{u}_{\text{in}}\boldsymbol{f}_{\text{in}}(t), \qquad \text{(Equation 6)}$$

where $\tau_v = 10$ ms. For computations that require external input a m_{in}-dimensional input, $\boldsymbol{f}_{\text{in}}(t)$, is applied to each neuron through a $N\times m_{\text{in}}$ matrix $\boldsymbol{u}_{\text{in}}$. $\boldsymbol{v}_\mu$ is a constant equilibrium potential. It is computed prior to learning to control the average firing rate of each neuron. (See section below.)

Learning w by RLS

The P-component vector $\boldsymbol{y}_{\text{targ}}(t)$ denotes the target factors, i.e., the factors we want the network to construct. We identify the elements of $\boldsymbol{w}$ by solving the following equation with a recursive least squares (RLS) algorithm,

$$\boldsymbol{w}\boldsymbol{s}(t) \approx \boldsymbol{y}_{\text{targ}}(t). \qquad \text{(Equation 7)}$$

We note that although we have specified the learning problem as requiring that we learn $\boldsymbol{w}$ and choose $\boldsymbol{u}$ randomly, this choice simply corresponds to a particular factorization of $\boldsymbol{J}_{\text{fac}}$ that makes the factors' location within the synaptic connectivity transparent. Learning would be identical if $\boldsymbol{J}_{\text{fac}}$ was identified by solving $\boldsymbol{J}_{\text{fac}}\boldsymbol{s}(t) \approx \boldsymbol{u}\boldsymbol{y}_{\text{targ}}(t)$.

Assuming a wisely chosen set of target factors (i.e., factors that can be learned), the network can perform computations that rely on the factors as a basis. The results of these computations take the form of network outputs. The m-component vector $\boldsymbol{f}_{\text{targ}}(t)$ denotes the desired (i.e., target) network outputs. Typical outputs are muscle EMGs (in the case where factors are derived from motor cortex) or the solution to an artificial computational problem. The network-generated output can be found by regression: $\boldsymbol{W}\boldsymbol{y}(t) + \boldsymbol{W}_0 \approx \boldsymbol{f}_{\text{targ}}(t)$. $\mathbf{W}_0$ is a bias term that is learned.

Connections that are not learned

The entries of $\boldsymbol{u}$, $\boldsymbol{u}_{\text{in}}$, and $\boldsymbol{J}_0$ are not modified by a learning procedure, and in most cases are selected randomly. (In cases where factors are derived from a firing rate model, we use knowledge of the rate model's connectivity to define $\boldsymbol{u}$ and $\boldsymbol{u}_{\text{in}}$. See section below.) The entries of $\boldsymbol{u}$ and $\boldsymbol{u}_{\text{in}}$ are selected randomly to reflect the fact that empirical neural responses often reflect random combinations of the empirical factors (which in turn presumably reflect both network dynamics and network inputs). The entries of $\boldsymbol{J}_0$ are selected randomly to capture, as simply as possible, aspects of connectivity related to tasks other than the current task mediated by $\boldsymbol{J}_{\text{fac}}$. This choice also has the desired effect of encouraging the network to generate irregular spiking.[52]

The elements of $\boldsymbol{u}$ and $\boldsymbol{u}_{\text{in}}$ are selected from a uniform distribution between -1.0 and 1.0, and then both are scaled by a scalar g which sets the overall scale of the factor and external input into each neuron. The elements of the first N columns of $\boldsymbol{J}_0$ are sampled from a Gaussian distribution with mean $\mu_{\text{f}}/N\tau_{\text{f}}$ and variance $g_{\text{f}}^2/N\tau_{\text{f}}^2$ and the elements of the second N columns are sampled from a Gaussian distribution with mean $\mu_{\text{s}}/N\tau_{\text{s}}$ and variance $g_{\text{s}}^2/N\tau_{\text{s}}^2$.

Setting the average spiking rate

$\boldsymbol{v}_\mu$ is N-dimensional vector of constants that sets the equilibrium potential for each spiking neuron (see Equation 6). It can be used to control each neuron's average rate of spiking. It is composed of three terms:

$$\boldsymbol{v}_\mu = v_{\text{rest}} + v_{\bar{\mu}} - \langle \boldsymbol{u}\boldsymbol{y}_{\text{targ}}(t) + (\boldsymbol{J}_0 - \langle \boldsymbol{J}_0 \rangle)\boldsymbol{s}(t) \rangle_T, \qquad \text{(Equation 8)}$$

where $\langle \boldsymbol{J}_0 \rangle$ is the average synaptic strength due to $\boldsymbol{J}_0$.

The first term, $v_{\text{rest}} = -10$ mV, determines the equilibrium potential of each neuron in the absence of input. The second term, $v_{\bar{\mu}} = 10$ mV, is a constant, global excitatory input that ensures the network generates activity in the absence of recurrent input.

The third term is computed prior to learning by applying the target factors as external inputs. Doing this mimics the effect they will have after learning at which point they will be constructed internally. $\langle \cdot \rangle_T$ indicates an average across a length of time T (typically around 100 trials per condition) over which this mean is computed.

The third term is equal to the temporally-averaged recurrent input each neuron receives. There are two sources of this recurrent input. The first is input due to the factors (which after learning will be constructed recurrently by the matrix $\boldsymbol{J}_{\text{fac}}$). The second is recurrent input due to $\boldsymbol{J}_0$. (Note that we are not subtracting the average temporal input due to the mean weights of $\boldsymbol{J}_0$, $\langle \boldsymbol{J}_0 \rangle$, because the strength of this average recurrent input has been selected to ensure that inhibition dominates in the network.) Although including the third term in $\boldsymbol{v}_\mu$ aids network construction by adjusting the total synaptic input to each neuron so that they all have a firing rate of roughly the same value, doing so is not strictly necessary.

For all examples, $\mu_{\text{f}} = -0.3$ and $\mu_{\text{s}} = 0$ (the mean parameters of $\boldsymbol{J}_0$) so that inhibition dominates. Strong recurrent inhibition offsets the strong external input $(v_{\bar{\mu}})$ that would otherwise cause high firing rates (given v_{rest}). One might wonder why strong recurrent

inhibition and strong external excitation were included, given they roughly cancel each other. Empirically, we observed that removing both in an untrained network eventually lead to network quiescence. Our choice of v_{rest}, $v_{\overline{n}}$ and μ_f sets the average firing rate across the network to approximately 15 spikes/second when averaged across time in a trial and across all task conditions.

Balancing task performance and spiking irregularity

The irregularity or regularity of spiking is set by the ratio of each neuron's synaptic input variance to its synaptic input mean. For example, consider multiple repeats of a cycle, as in Figure 2A. Spiking will be irregular across cycles if a neuron's net input displays large across-cycle variance despite a weakly modulated across-cycle mean. Spiking will be more regular if the mean input is strongly modulated throughout each cycle, and across-cycle variance is small. A larger $\boldsymbol{J}_{fac}$ leads to more regular spiking, because it creates a stronger factor-based input, and factor-based inputs have low across-cycle variance (Figure 3). In contrast, a larger $\boldsymbol{J}_0$ creates greater spiking irregularity because it increases the magnitude of the non-factor-based input, which is inconsistent across trials (Figure 3).

g_f and g_s set the scale of $\boldsymbol{J}_0$, while g sets the scale of $\boldsymbol{J}_{fac}$. In general, if the ratio of g_f and g_s to g is large, for a fixed population size, the network will produce more variable spike trains and perform less well on the task (though this cost of variable spiking may be small for large networks and when the number of factors is small). g must be sufficiently large to induce a sufficiently large mean fluctuation across the population to produce enough spikes to construct the factors (i.e., if the mean for all neurons was zero, constructing temporally fluctuating factors with $\boldsymbol{J}_{fac}$ would be impossible). Thus g, g_f, and g_s must be set to achieve two goals: 1) ensure that the synaptic input mean is large enough for learning to succeed; 2) ensure that the synaptic input variance is large enough to create realistic spiking variability. For all examples, $g_f = 0.13$ and $g_s = 0.11$ and g took values that ranged between 3 and 6, which we found produced spiking irregularity consistent with empirical observations (see Figure 6 and Churchland et al.[51]), while still enabling the network to construct the factors accurately.

Dividing the synaptic input into factor-based and non-factor-based components

Each neuron receives postsynaptic input due to the factors via $\boldsymbol{J}_{fac}$. It might initially seem that this is the only factor-based input each neuron receives. However, because $\boldsymbol{J}_0$ is a full-rank matrix it also contributes postsynaptic input to each neuron that is collinear with the factors. To see this, consider the activity of one pre-synaptic neuron and one post-synaptic neuron before and after training. Before training, the pre-synaptic neuron's spiking activity is completely unrelated to the factors, and thus its connections to the post-synaptic neuron through $\boldsymbol{J}_0$ convey no factor-based information. Post-training, the pre-synaptic neuron's spiking (noisily) reflects the factors, and thus so does its influence on the post-synaptic neuron via $\boldsymbol{J}_0$. Indeed, the post-synaptic neuron now receives, via $\boldsymbol{J}_0$, factor-related inputs from many pre-synaptic neurons and these do not necessarily sum to zero.

In principle one might wish to adjust $\boldsymbol{J}_0$ so that the total factor-based influence of $\boldsymbol{J}_0$ is zero. However, because of the nonlinear dependence between the factors and the network spiking activity and because of our online learning procedure, doing so would be challenging and would impose a structure on $\boldsymbol{J}_0$ that would cause it to no longer be random. Furthermore, conceptually there is nothing wrong with unmodified synapses contributing to the learned computation. Thus, instead of attempting to combat the contribution of $\boldsymbol{J}_0$ to the factor-based input, we simply compute it. After learning, we find the activity collinear with the factors that arises due to $\boldsymbol{J}_0$ and add this with the input that arises due $\boldsymbol{J}_{fac}$, yielding the total factor-based input into each neuron. The residual activity due to $\boldsymbol{J}_0$ is the non-factor-based component.

To find the factor-based input due to $\boldsymbol{J}_0$ we regress the synaptic input due to $\boldsymbol{J}_0$ against the factors, to identify a $N \times P$ dimensional matrix $\boldsymbol{u}_{J_0}$:

$$\boldsymbol{u}_{J_0}\boldsymbol{y}(t) \approx \boldsymbol{J}_0\boldsymbol{s}(t). \quad \text{(Equation 9)}$$

We define the factor-based input as the sum of the synaptic input that produces the factors, due to $\boldsymbol{u}$ and due to $\boldsymbol{J}_0$,

$$\boldsymbol{z}^F(t) = \left(\boldsymbol{u} + \boldsymbol{u}_{J_0}\right)\boldsymbol{y}(t) = \boldsymbol{u}'\boldsymbol{y}(t). \quad \text{(Equation 10)}$$

We define the non-factor-based component of the synaptic input as the residual:

$$\boldsymbol{J}_0\boldsymbol{s}(t) - \boldsymbol{u}_{J_0}\boldsymbol{y}(t), \quad \text{(Equation 11)}$$

which is equivalent to the total synaptic input minus the factor-based input.

In cases where external inputs are known, external inputs are effectively 'inherited' factors, and contribute to $\boldsymbol{z}^F(t)$, both due to their external synapses and due to $\boldsymbol{J}_0$. In such cases, we concatenate factors and external inputs into a $P + m_{in}$ dimension vector $\boldsymbol{y}'(t) = [\boldsymbol{y}(t), \boldsymbol{f}_{in}(t)]$. $\boldsymbol{y}'(t)$ thus captures both the factors generated internally by the network, and factors it inherits from upstream networks via its inputs. The factor-based input is then computed analogously to that above:

$$\boldsymbol{z}^F(t) = \left([\boldsymbol{u}, \boldsymbol{u}_{in}] + \boldsymbol{u}_{J_0}\right)\boldsymbol{y}'(t) = \boldsymbol{u}'\boldsymbol{y}'(t), \quad \text{(Equation 12)}$$

where $[\boldsymbol{u}, \boldsymbol{u}_{in}]$ is a $N \times P + m_{in}$ matrix and $\boldsymbol{u}_{J_0}$ is found by regressing the synaptic input due to $\boldsymbol{J}_0$ against $\boldsymbol{y}'(t)$. As above, the non-factor-based input is simply the total synaptic input minus the factor-based input.

Deriving target factors

Target factors can be derived from a number of sources, including neural recordings and mathematical models, using a variety of methods. In this study, we consider target factors derived from real and artificial neural populations. We denote the activity of the 'target-providing' population by the $\tilde{N}$ component vector $\boldsymbol{x}(t)$, where $\tilde{N}$ designates the number of neurons (real or artificial). $\boldsymbol{x}(t)$ describes the firing rate of every neuron in the population at time t. When deriving target factors from a rate-based network, firing rates are known because they are explicitly modeled. When deriving target factors from an empirically recorded population, firing rates are estimated (details below). In cases where there is only one behavioral condition (e.g., the cycling task), t indexes across times in that condition. In cases where there are multiple behavioral conditions, t indexes both across time and condition. Many of our rate networks were trained on multiple variants of each condition and t then also indexes across these as well.

For all our examples, we use principal component analysis to obtain the target factors. We learn a $P\times\tilde{N}$ matrix $\boldsymbol{V}$ so that $\boldsymbol{y}_{\text{targ}}(t)$ captures a specified fraction of the firing-rate variance of the target-providing population. To do so, we construct a $\tilde{N}\times T$ data matrix, where T is the number of times. We compute the $\tilde{N}\times\tilde{N}$ covariance matrix, followed by the eigenvectors of this matrix to yield the principal vectors, $\boldsymbol{V}$. From this,

$$\boldsymbol{y}_{\text{targ}}(t) = \boldsymbol{V}\boldsymbol{x}(t), \qquad \text{(Equation 13)}$$

yields the target factors.

In the case of neural recordings, we considered factors derived from single-unit electrophysiological recordings. However, target factors could in principle be derived from other types of neural recordings, such as multiunit recordings or calcium imaging data, provided that the data allow the factors to be accurately estimated. Other work has indicated that factors derived from multi-unit activity largely agree with those derived from single-units.[43] The same can be true of calcium imaging data, provided a sufficiently responsive Ca2+ indicator or appropriate pre-processing.[95]

Although we learn factors by principal component analysis throughout, other methods, such as factor analysis or LFADS, can be used and we found that factors derived using these other methods worked equally well as training targets. This is unsurprising; any linear dimensionality reduction method that accurately captures firing rate variance will yield factors that are similar up to a rotation. Our method should work similarly well for factors derived via any dimensionality reduction method, provided the factors do not exhibit 'pathologies' that would be challenging for any dynamical system to learn. For example, factors that exhibited high trajectory tangling[14] would be challenging to learn for any neural-network training method. In some situations, it may be more appropriate to summarize empirical data using non-linear dimensionality reduction techniques. Our training approach is currently not tailored to that situation (because network factors are linear combinations of spikes) but could potentially be modified to handle such situations. Lastly, although many dimensionality reduction methods ensure that the factors are orthogonal (i.e., uncorrelated across times and conditions), and encouraging the factors away from colinearity may aid training, orthogonality of the target factors is not strictly necessary.

Cycling task factors

Experimentally measured spike times from $\tilde{N} = 109$ neurons were collected from the motor cortex of primates performing a cycling task described in Russo et al.[14] The spike times were convolved with a Gaussian kernel (std. dev. = 25 ms) and trial-averaged. Trial averaging involved an 'adaptive alignment' procedure to align periods of the cycling movement; details of this procedure can be found in Russo et al.[14] For simplicity, we considered data from one condition: when the monkey pedaled forward for seven consecutive cycles.

Trial-averaged firing rates were 'soft normalized' following a procedure we have commonly used in the past.[14] This procedure is motivated by the finding that when using PCA, a common problem is that neurons with large firing-rate ranges can dominate (a neuron with a range of 100 spikes/s has 25 times as much variance as a neuron with a range of 20 spikes/s). For each neuron, we compute its firing rate range: the maximum trial-averaged rate (across all times in all conditions) minus the minimum. The normalization factor was this range plus a constant of 5 spikes/s. Adding the constant ensures that very low-rate neurons contribute less to the computation of the PCs than do high-rate neurons. Neurons with firing-rate ranges well above 5 spikes/s contribute roughly equally. Each neuron's activity (at all times and for all conditions) was normalized by this normalization factor:

$$x_n(c,t) \leftarrow \frac{x_n(c,t)}{5 + \text{range}_{c,t}(x_n(c,t))} \qquad \text{(Equation 14)}$$

The trial-averaged firing rates repeated across the middle cycles of the 7 cycle movement but were not exactly periodic because the activity of each neuron on each cycle did not end at exactly the same firing rate where it began. To simplify network training, we wished to construct a 'representative' single cycle that was perfectly periodic.[14] A simple solution is to set the representative cycle to be the trial-averaged activity of the middle cycle (cycle 4). Activity of cycle 4 (like the other middle cycles) repeated close to perfectly. Treating this activity as a periodic function created only a small discontinuity at the phase where activity 'wrapped around'. A reasonable solution would have been to simply smooth over this small discontinuity. However, the presence of multiple cycles allowed for a more elegant solution. The last half of the 4^{th} cycle had no discontinuity with the first half of the 5^{th} cycle. Thus, we created a representative cycle that began as the beginning of the 5^{th} cycle and continuously became more similar to the 4^{th} cycle (via a weighted average that began with weights of 1 and 0 and ended with weights of 0 and 1) until it exactly matched the 4^{th} cycle halfway through

that cycle. The representative cycle was then identical to the 4^{th} cycle for the second half. This produced a representative cycle that was perfectly periodic with no discontinuity.

Once the above processing was performed for each neuron, principal component analysis was performed on the population of firing rates and the projections of the firing rates onto the top 12 principal components were retained (See 'Deriving factor targets' subsection above for details on computing the PCs). Because the underlying activity of each neuron was periodic (see above) there was no discontinuity in each projection but there could still be a small discontinuity in first derivative. To ensure this was not the case, we concatenated 3 (perfectly repeating) cycles of the PC projections, smoothed each with a Gaussian kernel (std. dev. = 5 ms) and retained the middle cycle. This ensured well-behaved training targets, each of which was periodic in both its values and its first derivative. We concatenated four of these to create a 2 second 'trial'.

EMG recordings from the cycling task were high-pass filtered at 40 Hz and rectified. Then, they were smoothed, trial-averaged, soft-normalized, and processed, as described above, to obtain a closed, periodic loop.

Standard FORCE learning in rate models

FORCE learning considers a continuous-time recurrent neural network of firing rate units.[48] As a reminder, we denote the activity of the population by the $\tilde{N}$ component vector $\boldsymbol{x}(t)$. Units are connected through an $\tilde{N}\times\tilde{N}$ random matrix $\tilde{\boldsymbol{J}}$. The elements of $\tilde{\boldsymbol{J}}$ are selected independently from a Gaussian distribution of zero mean and variance $\tilde{g}^2/\tilde{N}$. A $m\times\tilde{N}$ matrix of output synapses $\tilde{\boldsymbol{W}}$ is modified by recursive least squares so that the network output matches a target output $\boldsymbol{f}_{\text{targ}}(t)$:

$$\tilde{\boldsymbol{W}}\boldsymbol{x}(t)\approx\boldsymbol{f}_{\text{targ}}(t). \qquad \text{(Equation 15)}$$

A $\tilde{N}\times m$ matrix of random synapses $\tilde{\boldsymbol{u}}$ carries the network's approximation of the target output back into the network, as though it were an input, augmenting the network's recurrent connectivity. The elements of $\tilde{\boldsymbol{u}}$ are selected from a uniform distribution between -1.0 and 1.0. When $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{J}}$ are appropriately scaled, the network is able to produce a good approximation to the target output because the approximation of the target is fed back into the network via $\tilde{\boldsymbol{u}}$ and is mixed with the recurrent feedback due to $\tilde{\boldsymbol{J}}$. This stabilizes the network dynamics, enabling learning.

After learning, the dynamics of $\boldsymbol{x}(t)$ are given by

$$\tau\frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t)+\varphi\left(\tilde{\boldsymbol{J}}\boldsymbol{x}(t)+\tilde{\boldsymbol{u}}\tilde{\boldsymbol{W}}\boldsymbol{x}(t)+\tilde{\boldsymbol{u}}_{\text{in}}\boldsymbol{f}_{\text{in}}(t)\right), \qquad \text{(Equation 16)}$$

where τ = 10 ms, $\varphi(\bullet)=\tanh(\bullet)$ is the nonlinear input/output function for each unit, and $\boldsymbol{f}_{\text{in}}(t)$ is an external input applied to each neuron by connections specified by a $\tilde{N}\times m_{\text{in}}$ matrix $\tilde{\boldsymbol{u}}_{\text{in}}$. The elements of $\tilde{\boldsymbol{u}}_{\text{in}}$ are selected randomly from a uniform distribution between -1.0 and 1.0 and are not modified. Once trained, by definition, the network produces a set of factors sufficient for constructing $\boldsymbol{f}_{\text{targ}}(t)$ because the activity $\boldsymbol{x}(t)$ can be linearly combined to produce $\boldsymbol{f}_{\text{targ}}(t)$ (i.e., Equation 15).

Modified FORCE learning & factors from the rate model

For the reaching task, we used a modified version of FORCE to train a rate network from which we could obtain target factors.[49] Somewhat surprisingly, the factors a FORCE trained network produces are identical to the factors the same network would produce if we simply apply $\boldsymbol{f}_{\text{targ}}(t)$ (normally a target output of the trained network) as an input through $\tilde{\boldsymbol{u}}$, without learning $\tilde{\boldsymbol{W}}$. To see this, consider the dynamics of $\boldsymbol{x}(t)$ in such a scenario:

$$\tau\frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t)+\varphi\left(\tilde{\boldsymbol{J}}\boldsymbol{x}(t)+\tilde{\boldsymbol{u}}\boldsymbol{f}_{\text{targ}}(t)+\tilde{\boldsymbol{u}}_{\text{in}}\boldsymbol{f}_{\text{in}}(t)\right). \qquad \text{(Equation 17)}$$

The only difference between Equations 17 and 16 is the second term within $\varphi(\bullet)$. Of course, when FORCE learning is successful, $\tilde{\boldsymbol{W}}\boldsymbol{x}(t)\approx\boldsymbol{f}_{\text{targ}}(t)$, so the second term will be almost identical in both cases. Thus, from the standpoint of the firing rates of rate-network units (and therefore of the factors derived from them), it matters little whether $\boldsymbol{f}_{\text{targ}}(t)$ is generated by the network's own recurrence (i.e., an output that is fed back in, as in standard FORCE training) or is simply applied to the network. Of course, without FORCE training the rate-based network cannot autonomously generate the factors. However, this is unnecessary for our purposes because we simply wish to know the patterns the rate network produces when it generates $\boldsymbol{f}_{\text{targ}}(t)$. Once those patterns are known we can use them as target factors to train the spiking network.

Therefore, we apply $\boldsymbol{f}_{\text{targ}}(t)$ as an input to the rate model, and only learn $\tilde{\boldsymbol{W}}$ to ensure that the factors are sufficient for reproducing $\boldsymbol{f}_{\text{targ}}(t)$, (i.e., we solve Equation 15 but do not feed back the output constructed by $\tilde{\boldsymbol{W}}$, as it is already being applied as an external input). After network training is complete, the target factors $\boldsymbol{y}_{\text{targ}}(t)$ are defined by Equation 13.

Backpropagation trained rate network & its factors

For the contextual integration task, we use a discrete time network, which we train with backpropagation through time. Network neurons evolve according to the following dynamics:

$$\boldsymbol{x}(t+\Delta t) = \varphi\left(\tilde{\boldsymbol{J}}\boldsymbol{x}(t) + \tilde{\boldsymbol{u}}_{\text{in}}\boldsymbol{f}_{\text{in}}(t) + \tilde{\boldsymbol{b}}\right), \quad \text{(Equation 18)}$$

where $\Delta t = 10$ ms and $\tilde{\boldsymbol{b}}$ is a time-independent bias input for each unit. Network output is defined as in Equation 15. $\varphi(\bullet)$ is defined as in the FORCE trained rate model. $\tilde{\boldsymbol{J}}$, $\tilde{\boldsymbol{u}}_{\text{in}}$, $\tilde{\boldsymbol{b}}$, and $\tilde{\boldsymbol{W}}$ are modified via backpropagation through time so that $\tilde{\boldsymbol{W}}\boldsymbol{x}(t) \approx \boldsymbol{f}_{\text{targ}}(t)$. After training is complete, the target factors $\boldsymbol{y}_{\text{targ}}(t)$ are defined by Equation 13.

We trained the firing rate model using the ADAM optimizer with standard decay rate parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and an initial step size of $\alpha = 0.001$. We performed 100 iterations of gradient descent and then reduced the step size by 2/3. The step size was reduced 10 times. Input, recurrent, and output synapses were initialized with Gaussian random variables, with standard deviation of $1/\sqrt{\tilde{N}}$, $0.9/\sqrt{\tilde{N}+1}$, and $1/\sqrt{\tilde{N}+1}$, respectively, where $\tilde{N}$ is the number of neurons ($\tilde{N} = 100$). Our optimization objective included a L2 penalty on the sum of the squared values of the input and recurrent synapses, scaled by a constant = 2e-6, to prevent the connections from growing too large.

Defining u and u_{in} when using rate-model derived factors

When training a spiking network using target factors derived from data, $\boldsymbol{u}$ (which determines how the factors are linearly combined into each spiking neuron) and $\boldsymbol{u}_{\text{in}}$ (which specifies how external inputs are weighted into each spiking neuron) are selected randomly. Random selection creates neurons whose firing rate reflects combinations of factors and external inputs without preference for any single factor or input. Because the analogous connections in trained rate models are likewise effectively random (despite being learned), their firing rate units will also exhibit this property (unless learning is designed to specifically discourage it). To ensure robust spiking-network training, when deriving target factors from a trained rate model we sought to preserve the precise structure present in the trained rate model within the trained spiking model. While preserving properties between the two models was not critical (i.e., spiking network training could still succeed if this was not done), it likely made training robust especially for smaller networks.

We sought to preserve two key aspects from the trained rate model when training the spiking model: 1. the relative magnitudes of recurrent and external synaptic inputs; 2. the relationships between firing rates and inputs that emerged during rate-network training. As an example of aspect one, if rate-network training produces recurrent connectivity that is strong relative to incoming commands, we wish that to be preserved in the spiking model. As an example of aspect two, consider the preparatory epoch during the reaching task. A neuron's 'directional preference' (and thus which factors it reflects) is likely to be related to which external inputs it receives. A neuron that responds most before rightward reaches is likely to receive a contribution from the network-input that conveys the cosine of the reach angle.

Achieving the above goals would be simple if rate networks and spiking networks had the same number of units. One could simply pick $\boldsymbol{u}$ and $\boldsymbol{u}_{\text{in}}$ so that each spiking neuron had a factor-based synaptic input and an 'external' synaptic input approximating that of a corresponding rate unit. However, spiking networks typically had more neurons than the corresponding rate network had units. We thus developed a procedure that yields $\boldsymbol{u}$ and $\boldsymbol{u}_{\text{in}}$ that are each 'internally' random (i.e. each matrix is itself random) but have relative magnitudes that mirror what occurred in the rate network, and are also 'aligned' in the sense that spiking neurons will reflect combinations of factors and external inputs in a manner similar to that for rate-model neurons. The outcome of this procedure is a spiking network that can be larger than the rate network upon which it is based, but where neurons have overall similar response properties.

First, we defined a $N \times \tilde{N}$ matrix $\boldsymbol{U}$ with orthonormal columns to specify a linear map between the dimension of the rate model ($\tilde{N}$) and the dimension of the spiking model (N). The elements of this matrix were chosen randomly and then its columns were orthonormalized. To ensure the scale of the recurrent factor feedback and external input into each spiking neuron matches (overall) that of the recurrent factor feedback and external input into each rate unit, we leveraged knowledge of the recurrent and external connectivity of the rate network, $\tilde{\boldsymbol{J}}$ and $\tilde{\boldsymbol{u}}_{\text{in}}$ respectively. Recalling that $\boldsymbol{V}$ is the $\tilde{N} \times P$ matrix that defines the factors from the activity of the rate units, we define $\boldsymbol{u} = g\boldsymbol{U}\tilde{\boldsymbol{J}}\boldsymbol{V}^{\text{T}}$. Recalling that $\boldsymbol{u}$ is multiplied by $\boldsymbol{y}(t)$ in the spiking network, we can consider the impact of each term: $\boldsymbol{V}^{\text{T}}$ maps the factors into the $\tilde{N}$-dimensional rate network space, $\tilde{\boldsymbol{J}}$ then maps the factors to synaptic inputs in the $\tilde{N}$-dimensional rate-network space (preserving the scale present within the trained rate network), and $\boldsymbol{U}$ maps those preserved values into the N-dimensional space of the spiking network, preserving the norm. To accomplish something similar for external inputs, $\boldsymbol{U}$ is also used to define $\boldsymbol{u}_{\text{in}}$: $\boldsymbol{u}_{\text{in}} = g\boldsymbol{U}\tilde{\boldsymbol{u}}_{\text{in}}$. Defining $\boldsymbol{u}_{\text{in}}$ based on $\tilde{\boldsymbol{u}}_{\text{in}}$ has the same effect that defining $\boldsymbol{u}$ based on $\tilde{\boldsymbol{J}}$ does. Note that both $\boldsymbol{u}_{\text{in}}$ and $\boldsymbol{u}$ are scaled by g, as when they were completely random, to set their overall scale. The above procedure ensures that both $\boldsymbol{u}$ and $\boldsymbol{u}_{\text{in}}$ are composed of random matrices (e.g. $\boldsymbol{U}$, $\tilde{\boldsymbol{u}}_{\text{in}}$, and $\tilde{\boldsymbol{J}}$) and thus are themselves random, despite being designed through an intentional procedure.

Quantifying performance, factor-based input variability, and spiking irregularity in trained spiking models

To determine how accurately the factors matched their targets, we computed the normalized mean-squared error. For the r^{th} trial the normalized error is

$$E_r = \frac{\langle \left(\boldsymbol{w}\boldsymbol{s}(t) - \boldsymbol{y}_{\text{targ}}(t)\right)^2 \rangle_{T,P}}{\langle \boldsymbol{y}_{\text{targ}}(t)^2 \rangle_{T,P}}, \quad \text{(Equation 19)}$$

where T is time spanned by the trial for tasks with a trial structure. P indexes the factors. We report the across-trial median of this error to provide an aggregate performance measure for tasks where performance can vary between trials of different conditions.

To determine the across-trial variability of the factor-based synaptic input for neuron n, we computed a normalized across-trial variance using the following equation:

$$\frac{\left\langle \mathrm{Var}\left(z^F_{r,n}\left(t\right)\right)_R \right\rangle_T}{\left\langle \mathrm{Var}\left(z^F_{r,n}\left(t\right)\right)_T \right\rangle_R}, \quad \text{(Equation 20)}$$

for R trials of duration T. The numerator captures across-trial variability, while the denominator captures the degree to which the mean varies across time. For pure noise, this expression would be unity on average. Small values indicate that the systematic aspect of synaptic input is large relative to across-trial variability. Variability of network outputs, the factors, and the non-factor based synaptic inputs were computed using the same expression. For networks where there was more than one condition, values were averaged across conditions.

To determine how variable spiking was from trial to trial, we calculate the 'across trial' Fano factor following the method established in Churchland et al.[80] For a given neuron and condition, for each trial we counted the number of spikes in a window from time t to $t + \Delta t$. We computed the across-trial mean and across-trial variance of the count. For all analyses, $\Delta t = 100$ ms and t was incremented in 10 ms steps. This yielded one value of the spike-count mean and one value of the spike-count variance for every neuron, condition, and time-window.

To summarize the Fano factor for a given neuron, we found the slope of the best fit line (constrained to pass through zero) describing the relationship between the spike-count variance and the spike-count mean across all conditions and time-windows. To summarize the Fano factor for a population of neurons, we did the same but with one point per neuron, condition and time-window.

Computing the flow field of the factors

To compute the flow field in the factor basis (Figure 1C) we perturbed the first two factors at various points along a reference trajectory and observed how the network state relaxed following the perturbation. We first computed the average factor trajectory (for all 12 factors) across multiple trials in order to compute the direction of the approximate derivative at each point along the average trajectory. We selected perturbations of the first two factors that were orthogonal to the average trajectory and observed the network state 10 ms later to compute the flow field in the vicinity of the stable factor limit cycle.

Computing $r^F_n(t)$

To define a firing rate for each neuron, we modeled the relationship between the factor-based synaptic input and the probability of spiking. Intuitively, as the factor-based synaptic input increases, the probability of spiking also increases. The relationship is probabilistic because the neuron also receives a sizeable non-factor-based input. If one knows only the current factor-based input, then the best one can do is predict the probability of a spike occurring. For LIF neurons, in principle there is a monotonically-increasing nonlinear function that describes this relationship.[93] However, analytically deriving this function was beyond the scope of this work. We took the alternative approach of approximating it. This approximation required identifying two parameters for each neuron, b^1_n and b^0_n.

We define the factor-based firing rate for the n^{th} neuron as

$$r^F_n(t) = \exp\left(b^1_n z^F_n(t) + b^0_n\right). \quad \text{(Equation 21)}$$

b^1_n is a gain parameter that accounts for attenuation of the synaptic input by membrane properties (by τ_v for example). b^0_n is a constant offset parameter that accounts for the constant inputs each neuron receives (e.g., from $\boldsymbol{v}_\mu$) and also reflects the value of the spike threshold. In principle, these parameters could be set based on first principles, but doing so does not guarantee an accurate fit to the data because the approximate nonlinearity (i.e., the exponential function) is not identical to the true non-linearity. To circumvent this issue, we learned these parameters using standard optimization techniques.

Because the goal of a firing rate is to account for the probability of spiking in a given window, we model the spikes occurring between time t and Δt, $N_n(t)$, with a Poisson distribution:

$$N_n(t) \sim \mathrm{Pois}\left(r^F_n(t)\Delta t\right). \quad \text{(Equation 22)}$$

Spiking in our networks is only approximately Poisson, but this noise model was sufficient for the practical purpose of fitting an exponential non-linearity that captured the relationship between the factor-based input and the probability of spiking. Furthermore, choice of the Poisson distributions accords with the exponential function; the exponential is the canonical inverse link function for the Poisson distribution when fitting a generalized linear model (GLM).[55]

To learn b^0_n and b^1_n we use gradient ascent to maximize the log-likelihood of $N_n(t)$ with respect to b^0_n and b^1_n for all times (up to T, where T is the length of collected simulated data). The log-likelihood function for the Poisson distribution is

$$\mathcal{L}\left(b_n^0, b_n^1\right) = \sum_{t=0}^{T} N_n(t)\log\left(r_n^F(t)\Delta t\right) - r_n^F(t)\Delta t. \quad \text{(Equation 23)}$$

Optimizing this function can be performed with standard optimization packages. For our analyses, we used simulated network-generated factors and spikes from our cycling task, with Δt = 1 ms. Analyses were performed with built-in MATLAB functions.

LFADS analysis

Latent factor analysis via dynamical systems (LFADS) was performed as described in Pandarinath et al.[11] using code provided by the authors (https://github.com/google-research/computation-thru-dynamics). LFADS is a statistical model for inferring latent variables that can account for neural activity patterns. A salient feature of neural responses is their spatial and temporal co-variation, and LFADS accounts for this by assuming the data are generated by a recurrent neural network. Approximate inference is performed using a set of variational autoencoders. Spike trains were binned in non-overlapping 10 ms bins for this analysis.

LFADS offers the flexibility to infer exogenous inputs into the neural network to describe the spike train data if variations in the data cannot be sufficiently explained by variations in the initial conditions of the dynamics of an autonomous network. The results presented here inferred these additional inputs, but doing so only quantitatively changed the findings; qualitatively our results were unchanged. Because there actually were no exogenous inputs into the spiking network that generated the data for fitting LFADS, it seems likely that these learned inputs were actually the trial-to-trial fluctuations caused by the non-factor-based input. Resolving this requires further study.

Tangling analysis

Tangling analysis was performed as described in Russo et al.[14] using code provided by the authors (https://github.com/aarusso/trajectory-tangling). The method seeks to identify pairs of network states that have different derivatives yet are near one another. The pair of states can be either at different times within a single trial or at different moments in different trials. Tangling is computed as

$$Q(t) = \max_{t'} \frac{\left\|\dot{\boldsymbol{X}}_t - \dot{\boldsymbol{X}}_{t'}\right\|^2}{\left\|\boldsymbol{X}_t - \boldsymbol{X}_{t'}\right\|^2 + \varepsilon}, \quad \text{(Equation 24)}$$

where $\boldsymbol{X}_t$ is a state variable of interest (e.g. factors, synaptic input, etc.) at time t and $\dot{\boldsymbol{X}}_t$ is the derivative of that variable at time t (t indexes across all times in all conditions and/or trials being analyzed). If the value of this state variable is the same at two times but its derivative is different, an autonomous dynamical system cannot account for the observed dynamics. In practice it would be rare for the state of a spiking network (or data) to ever be truly identical at two times, but high tangling still implies that the observed state trajectories would be difficult to instantiate in a noise-robust autonomous dynamical system.

We assessed trajectory tangling of the factors, the factor-based input, the non-factor-based input, and the total synaptic input on single trials. Default options were applied based on previous analysis, including normalization of the signals (divide by their range, then add an offset) before applying the analysis. Contrary to prior analyses, PCA was not applied to the full state before analysis (e.g. for the factor-based input, the non-factor-based input, and the total synaptic input tangling was assessed in the full 1200 dimensional space). When tangling was compared between the factors and the network outputs, signals were filtered with a Gaussian kernel (5 ms s.d.) to conform to prior analyses[14] where filtering was applied.

Learning sparsely connected and Dale's Law obeying networks

To train networks while constraining $\boldsymbol{J}$ to be sparse and with columns of consistent sign, we developed a two-step method. Constraining $\boldsymbol{J}_0$ to abide by these constraints is straightforward, since its elements are chosen randomly and not modified; we sample these elements from a truncated Gaussian distribution to obtain exclusively excitatory or inhibitory elements, respectively, and set some connections equal to zero to achieve the desired sparsity.

In contrast, enforcing these constraints on $\boldsymbol{J}_{\text{fac}}$ is challenging from a training perspective.[96] The recursive least-squares (RLS) algorithm is an important part of the network construction; RLS ensures that the trained network is stable because the samples generated during the training not only characterize the desired activity, they also include typical fluctuations that arise during network operation.[48] Unfortunately, the RLS algorithm is impractical for weights that are constrained in sign. Sign constraints can easily be enforced in a batch least-squares (BLS) approach but using BLS for building recurrent network models does not guarantee that the resulting network state is stable.

We develop a procedure that combines the sampling property of RLS with the constraint amenability of BLS. First, we train a fully connected and sign-unconstrained spiking network (our standard training paradigm) using RLS to obtain the recurrent connectivity matrix $\boldsymbol{J}_{\text{fac}}$ that constructs the network factors. Then, we sample the post-synaptic inputs $\boldsymbol{J}_{\text{fac}}\boldsymbol{s}(t)$ over an extended period of time and use them to train post-synaptic inputs $\boldsymbol{J}^C_{\text{fac}}\boldsymbol{s}(t)$ of a second network with constrained weights (indicated by the superscript C). The least-squares problem for matching these two sets of inputs

$$\boldsymbol{J}^C_{\text{fac}}\boldsymbol{s}(t) \approx \boldsymbol{J}_{\text{fac}}\boldsymbol{s}(t), \quad \text{(Equation 25)}$$

is done in batch. This way, the data used to solve the least squares problem are assured to contain the fluctuations necessary for learning a stable solution and constraints can be easily enforced.

To illustrate why this two-step training procedure was necessary we sampled the deviations between the target factors and the actual factors for the fully connected and sign-unconstrained network trained with RLS, i.e.

$$\boldsymbol{\eta}^{\mathrm{RLS}}(t) = \boldsymbol{w}\boldsymbol{s}(t) - \boldsymbol{y}_{\mathrm{targ}}(t), \qquad \text{(Equation 26)}$$

for some length of time T (Figures 5D and 5E). We shuffled the temporal and spatial indices of $\boldsymbol{\eta}^{\mathrm{RLS}}(t)$, which we call $\boldsymbol{\eta}^{\mathrm{shuff}}(t)$, and used these shuffled errors to train the second network with constrained weights, as we did above, but instead according to the least-squares problem

$$\boldsymbol{J}^{\mathrm{C}}_{\mathrm{fac}}\boldsymbol{s}(t) \approx \boldsymbol{u}\left(\boldsymbol{y}_{\mathrm{targ}}(t) + \boldsymbol{\eta}^{\mathrm{shuff}}(t)\right). \qquad \text{(Equation 27)}$$

Here, $\mathbf{s}(t)$ is the spiking network activity when $\boldsymbol{y}_{\mathrm{targ}}(t)$ and $\boldsymbol{\eta}^{\mathrm{shuff}}(t)$ are applied as external input via $\boldsymbol{u}$.

Although the above procedure was effective at learning $\boldsymbol{J}^{\mathrm{C}}_{\mathrm{fac}}$ with the desired sparsity and sign constraints, it was not able to ensure that the sparsity patterns of $\boldsymbol{J}^{\mathrm{C}}_{\mathrm{fac}}$ and $\boldsymbol{J}_0$ matched. BLS assigns roughly half of the synaptic connections to zero, because it would prefer to make those connections negative (or positive) but cannot. We cannot pre-identify which connections this will apply to (and thereby apply the same sparsity pattern to $\boldsymbol{J}_0$). Instead, we must set the actual sparsity requirement of BLS to half the desired sparsity, and cannot achieve control over which synapses will actually be non-zero. However, we stress that, as we have shown, only $\boldsymbol{J}^{\mathrm{C}}_{\mathrm{fac}}$ provides the critical substrate for the factors (no new factors arise due to $\boldsymbol{J}_0$), implying that this mismatch is not likely to impact our results.

$\boldsymbol{J}^{\mathrm{C}}_{\mathrm{fac}}$ cannot be factorized into $\boldsymbol{w}$ and $\boldsymbol{u}$ as was the case for $\boldsymbol{J}_{\mathrm{fac}}$. Thus, this process replaces the low-rank matrix $\boldsymbol{J}_{\mathrm{fac}}$[46] with a full-rank matrix $\boldsymbol{J}^{\mathrm{C}}_{\mathrm{fac}}$ (since it is now a sparse and sign-constrained matrix). Because of this, $\boldsymbol{w}$ (which serves, in this case, purely to read out the network activity) must be learned separately from $\boldsymbol{J}^{\mathrm{C}}_{\mathrm{fac}}$ to extract the factors $\boldsymbol{y}(t)$ themselves (not the synaptic inputs that arise from them). This can readily be done with RLS or BLS after learning $\boldsymbol{J}^{\mathrm{C}}_{\mathrm{fac}}$.

Task details

Cycling task

We included a brief input pulse at the beginning of each period to initialize a trial and to compensate for phase drift than can accumulate when learning periodic tasks; results on this task are qualitatively similar if the pulse was not included. $m_{\mathrm{in}} = 1$. $\boldsymbol{f}_{\mathrm{in}}(t)$ was of amplitude 2.0 and duration 50 ms. $\boldsymbol{V}$ was calculated to capture 99% of the variance of $\boldsymbol{x}(t)$, yielding 12 factors $(P = 12)$. 800 model spiking neurons were used $(N = 800)$. The spiking network gain was $g = 4$.

Sparsely connected, Dale's Law cycling task

$\boldsymbol{f}_{\mathrm{in}}(t)$, N, P and g defined as for the fully connected cycling network. The elements of $\boldsymbol{J}_0$ were drawn such that 50% of the population was exclusively excitatory or inhibitory $(p_{\mathrm{EI}} = 0.5)$ and such that each neuron was only connected to 40% of the population $(p_{\mathrm{sparse}} = 0.4)$.

Reaching task

A two-dimensional input indicated the target direction and the termination of a step-input indicated the 'go cue'. The go cue occurred approximately 150 ms before reach onset and approximately 50 ms before the onset of changes in muscle activity (EMG). The direction-specifying input was present for the duration of the trial. The amplitude of each component of the direction-specifying input was $\frac{1}{2}\cos(\theta)$ and $\frac{1}{2}\sin(\theta)$, where θ specifies reach direction. These inputs were applied both to the spiking network and to the rate model used to identify the target factors.

Both networks – spiking and rate – had to produce empirical patterns of muscle activity. The rate model was trained to generate those output targets in order to identify target factors that could be used for spiking-network training. Assuming the rate network successfully generated the output targets, it should supply target factors that will also allow the spiking network to do so. To produce appropriate output targets for rate-network training, muscle activity was temporally filtered and windowed such that any (already minimal) changes in activity before the go cue were zero. Small variance modes (accounting for less than 1% of the total variance) of each muscle's EMG were removed and each muscle's EMG was normalized across all movement conditions so that its maximum amplitude was 2.0. We used pre-processing to create idealized targets because we did not wish the network to attempt to fit small idiosyncratic aspects of the empirical recordings. Details regarding the recording of EMG data can be found in Elsayed et al.[74]

During rate-network training, we included an additional target output with a constant value of -0.5 for all times. This aided rate-network training because it helped suppress chaotic fluctuations due to strong internal recurrence. When in a highly chaotic regime, rate models become less sensitive to inputs and therefore poorly reflect those inputs and input-derived signals necessary for successful training. This additional input shifted the network away from a chaotic regime, while maintaining strong internal recurrence, a regime better for learning. This output was not necessary when training the spiking network and was not used for that network. To mimic standard paradigms in experimental neuroscience, the time between trials and the sequence of trial conditions were chosen randomly when training both networks. The inter-trial interval duration was sampled from an exponential distribution with a mean of two seconds to which a minimum interval of 0.4 seconds was added.

The rate model was trained with our modified FORCE procedure, as described above. $\boldsymbol{V}$ was calculated to capture 99% of the variance of $\boldsymbol{x}(t)$, yielding 37 factors $(P = 37)$. 1200 spiking neurons were used $(N = 1200)$ and 800 rate units were used $(\tilde{N} = 800)$. The spiking network gain was $g = 3$ and the rate network recurrent gain was $\tilde{g} = 1.4$.

Contextual integration task

We simulated each of the two 'sensory' inputs to both the firing rate network and the spiking network as a constant plus zero-mean white noise. The mean of each sensory input for each trial was selected randomly from a uniform distribution from -0.1 to 0.1. Gaussian noise with std = 0.4 in each time step of Δt = 1 ms was added to the mean input so that the behavior of the discrete-time network roughly matched experimental subjects. Two additional 'context' inputs identified which of the two sensory inputs should be integrated (the 'cued' input) on the current trial. For example, when the first context input had a value of one and the second context input had a value of zero, the first sensory input was cued and the second sensory input should be ignored ('uncued'). The context input was present for the duration of a trial. In total, the network received four external inputs $(m_{\text{in}} = 4)$.

The rate model was trained with backpropagation through time, as described above. Performance of the spiking was judged by comparing the sign of the output at the end of the trial to the sign of the mean value of the cued input, with matching signs indicating a correct response. We varied the trial difficulty by changing the absolute magnitude of the mean value of each input, holding the noise constant. $\boldsymbol{V}$ was calculated to capture 95% of the variance of $\boldsymbol{x}(t)$, yielding 9 factors $(P = 9)$. 800 spiking neurons were used $(N = 800)$ and 100 rate units were used $(\tilde{N} = 100)$. The spiking network gain was $g = 3$.

Two-task network

To illustrate the potential of our approach to train networks on multiple tasks, and to provide insight into how factors from different tasks could be arranged in state space, we trained a network of 1200 LIF neurons to perform the cycling task of Figure 2 and the reaching task of Figure 6. The cycling factors were from data, as before, and the reaching factors were from a rate network, as before. All network parameters are identical to the reaching network of Figure 6, except g (the scaling of the learned recurrent feedback); $g = 6$ so that the learned recurrent inputs in the spiking network are slightly stronger than in prior networks.

On each trial, the network had an equal probability of performing either task. Which task was to be performed was indicated by which inputs were active. For the cycling task, a second 'stop' pulse was included so the network would stop cycling before the start of the next trial (not necessary in the original task). Inputs for the reaching task were as in Figure 6: a two-dimensional direction signal and a 'go' cue.

Only a single EMG output (the activity of the posterior deltoid) was used as an output target for the trained spiking network (i.e. after the factors were learned by the spiking network), because this was the only common EMG recording across both tasks. All EMGs were used in the initial rate model training for the reaching task (as in Figure 6) to derive the reaching-task factors, and the cycling task factors did not require knowledge of the EMGs because they were directly available.

12 cycling factors and 37 reaching factors were used to train the network. To illustrate the potential to design networks that can perform different tasks using factors that reside in different areas in the neural state space, the network was designed such that the factors for each task were roughly orthogonal to each other. To do this, we enforced that the cycling factors were zero during reaching tasks and the reaching factors were zero during the cycling task. To quantify this, we computed the angle between the subspace defined by the reaching factor projection of the neural activity (the first 37 columns of **w**) and the subspace defined by the cycling factor projection (the last 12 columns of **w**) using the MATLAB function *subspace*, and found that it was 85 degrees, i.e., roughly orthogonal.