nature computational science

Article

https://doi.org/10.1038/s43588-024-00688-3

Automated customization of large-scale spiking network models to neuronal population activity

Received: 22 September 2023

Accepted: 8 August 2024

Published online: 16 September 2024

Check for updates

Shenghao Wu[1,2,3], Chengcheng Huang[3,4,5], Adam C. Snyder[6], Matthew A. Smith[1,3,7,11], Brent Doiron[8,9,11] & Byron M. Yu[1,3,7,10,11]

Understanding brain function is facilitated by constructing computational models that accurately reproduce aspects of brain activity. Networks of spiking neurons capture the underlying biophysics of neuronal circuits, yet their activity's dependence on model parameters is notoriously complex. As a result, heuristic methods have been used to configure spiking network models, which can lead to an inability to discover activity regimes complex enough to match large-scale neuronal recordings. Here we propose an automatic procedure, Spiking Network Optimization using Population Statistics (SNOPS), to customize spiking network models that reproduce the population-wide covariability of large-scale neuronal recordings. We first confirmed that SNOPS accurately recovers simulated neural activity statistics. Then, we applied SNOPS to recordings in macaque visual and prefrontal cortices and discovered previously unknown limitations of spiking network models. Taken together, SNOPS can guide the development of network models, thereby enabling deeper insight into how networks of neurons give rise to brain function.

Computational models help us understand brain function by reproducing specific aspects of brain activity. Single-neuron models have provided a mechanistic foundation for the generation of action potentials[1]. Small neural circuit models, such as the stomatogastric ganglion (STG) model of crustaceans[2], have been used to understand the generation of rhythmic motor patterns. At the systems level, large-scale network models, including rate-based recurrent neural networks[3–5] and convolutional neural networks[6], have informed how neural circuits perform complex brain computations. Although neurons communicate through temporally complex spike trains, these network models focus on replicating neuronal firing rates without spikes. To better link computational models and biological spiking neurons, large-scale spiking neural networks (SNNs) have been proposed. These SNNs aim to produce population spike trains whose time course and/or variability mimic that of neuronal recordings[7–12]. SNNs are increasingly important large-scale models in computational neuroscience: studying mechanisms of the biologically realistic circuit of a SNN is a critical step in understanding complex processing in cortical circuits.

A key goal in constructing network models is to customize their parameters to recapitulate some aspect of the recorded neuronal activity. In single-neuron models and small neural circuit models, each model parameter corresponds to a specific biological component

[1]Neuroscience Institute, Carnegie Mellon University, Pittsburgh, PA, USA. [2]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA. [3]Neural Basis of Cognition, Pittsburgh, PA, USA. [4]Department of Neuroscience, University of Pittsburgh, Pittsburgh, PA, USA. [5]Department of Mathematics, University of Pittsburgh, Pittsburgh, PA, USA. [6]Department of Brain and Cognitive Sciences, University of Rochester, Rochester, NY, USA. [7]Department of Biomedical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. [8]Department of Statistics, University of Chicago, Chicago, IL, USA. [9]Grossman Center for Quantitative Biology and Human Behavior, University of Chicago, Chicago, IL, USA. [10]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. [11]These authors contributed equally: Matthew A. Smith, Brent Doiron, Byron M. Yu. e-mail: byronyu@cmu.edu

and can often be measured experimentally[1,2]. Larger-scale models, including rate networks and SNNs, are more difficult to customize because of the larger parameter space that comes with a larger number of neurons[13,14]. Furthermore, the lack of a one-to-one correspondence between each neuron in the model network and each recorded neuron challenges the comparison between model activity and neuronal recordings[15]. In particular, the number of neurons within the model network is often far smaller than the biological network that the model is intended to describe.

Different approaches have been used to circumvent the need for a one-to-one correspondence between model neurons and recorded neurons. One approach is to construct network models whose output, such as limb movement[9,16] or decisions[17], reproduces subject behavior given a network input. Such models are customized by optimizing a cost function representing the difference between model output and behavior. These network models have shown impressively similar activity features to activity recorded in the brain, albeit without explicit matching of neuronal activity in the cost function. The cost function can be optimized using methods such as first-order reduced and controlled error (FORCE)[13] or backpropagation[18] because it has a closed-form expression with respect to model parameters.

Another approach is to reproduce statistical measures of neuronal activity. SNNs are often designed to reproduce variability in individual neuron activity (for example, Fano factor (ff) of spike counts[19–22]) and pairwise spike count correlation[14,23–25]. SNNs have also been designed to reproduce population-wide covariability in neuronal recordings[26]. The cost function, representing differences in spiking activity statistics between the model and recordings, has no closed-form expression with the model parameters because it depends on computationally demanding numerical simulations and cannot be directly evaluated. So far, the parameters of these SNNs have been hand tuned[26], customized using exhaustive search[14,27] or customized using Bayesian deep-learning approaches when the network simulation time is small[28]. This has limited the exploration and understanding of the full range of activity regimes that large-scale SNNs are capable of exhibiting.

For example, different activity regimes have been identified for the classical balanced network (CBN)[7,29,30], the most widely studied SNN (Fig. 1a). Searching the high-dimensional parameter space to find a set of parameters that produces spike trains with specified properties is difficult. Simulating networks using all possible combinations of parameters can be computationally intractable due to the exponential growth in combinations of parameters. Furthermore, it is unknown a priori whether there even exists a combination of parameters (referred to as a parameter set) that produces spiking activity with the specified properties. Hence there is a clear need for an automated framework to search the parameter space.

We propose an automatic framework, Spiking Network Optimization using Population Statistics (SNOPS), for customizing the parameters of a large-scale SNN to reproduce observed spiking activity statistics. SNOPS uses Bayesian optimization (BO) to determine model parameters, a technique widely applied in machine learning for optimizing cost functions without a closed-form expression. We include population-wide activity statistics based on dimensionality reduction to obtain a closer match of the network model to neuronal recordings than using statistics defined only on individual neurons and pairs of neurons[31,32]. SNOPS provides a guided search of the parameter space and can help accelerate the development of SNNs, especially in settings where the network simulation time is large.

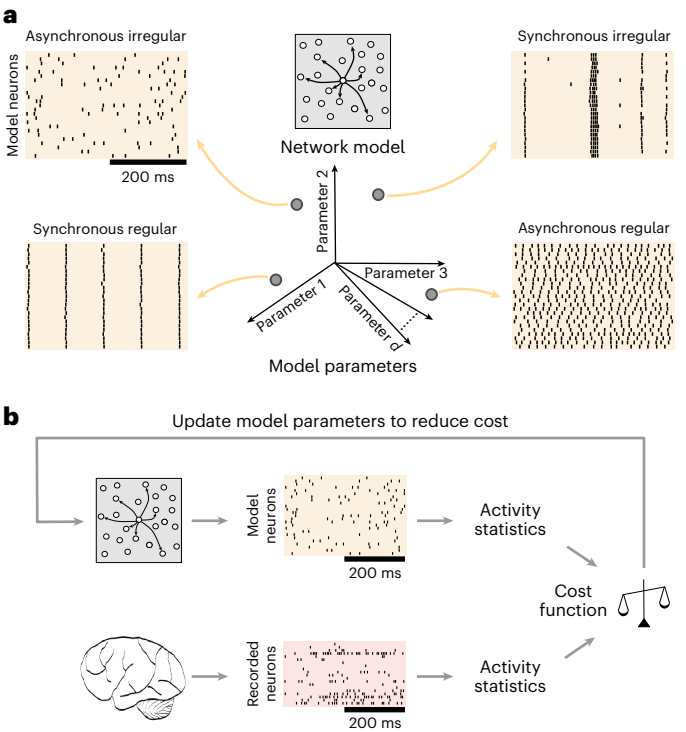

Fig. 1 | Framework for automated customization of a spiking network model to neuronal recordings. a, A SNN has a complicated dependency between its parameters and spiking output. For example, different parameter sets correspond to each of four previously identified activity regimes of a CBN: asynchronous irregular, synchronous regular, synchronous irregular and asynchronous regular. In this work, the SNN has eight parameters, including those that govern the connection strength between neurons as well as the timescale of synaptic decay (Supplementary Table 1). **b**, Our customization framework matches activity statistics of spike trains produced by the network model to those of neuronal recordings. It uses a guided searching algorithm to iteratively update the model parameters. The activity statistics are defined by the user and can include single-neuron, pairwise and population activity statistics.

Results

Model overview

SNOPS is designed to automatically customize a SNN to neuronal recordings (Fig. 1b). It is based on iteratively updating the model parameters to improve the correspondence between the spike trains generated by the network model and the recorded spike trains (using a cost function, defined below). The cost function is based on a set of activity statistics, which are computed for both the generated and recorded spike trains.

In the following sections, we first introduce the activity statistics used in this work and the SNOPS optimization framework. We next validate SNOPS using simulated activity. We then apply SNOPS to customize the CBN and its extension, the spatial balanced network (SBN)[11,26], to macaque visual area V4 and prefrontal cortex (PFC) recordings. We reveal that SBNs are better suited to reproduce key aspects of neuronal recordings than CBNs, and identify the specific combinations of activity statistics that the CBN cannot capture well.

Activity statistics for comparing models with neuronal activity

We first introduce the activity statistics used to compare the spike trains produced by the network model and the recorded spike trains. For all activity statistics, we begin by counting spikes within predefined time bins (Fig. 2a, left). This activity from individual neurons recorded simultaneously can be represented in a population activity space, where each axis represents the activity level of one neuron (Fig. 2a, center). We then compute activity statistics based on individual neurons, pairs of neurons and populations of neurons (Fig. 2a, right), as described below.

We considered two single-neuron statistics: the mean firing rate (fr) and ff (Fig. 2b). The mean fr is defined as the average level of activity across all neurons in the population and across all time bins. The ff

https://doi.org/10.1038/s43588-024-00688-3

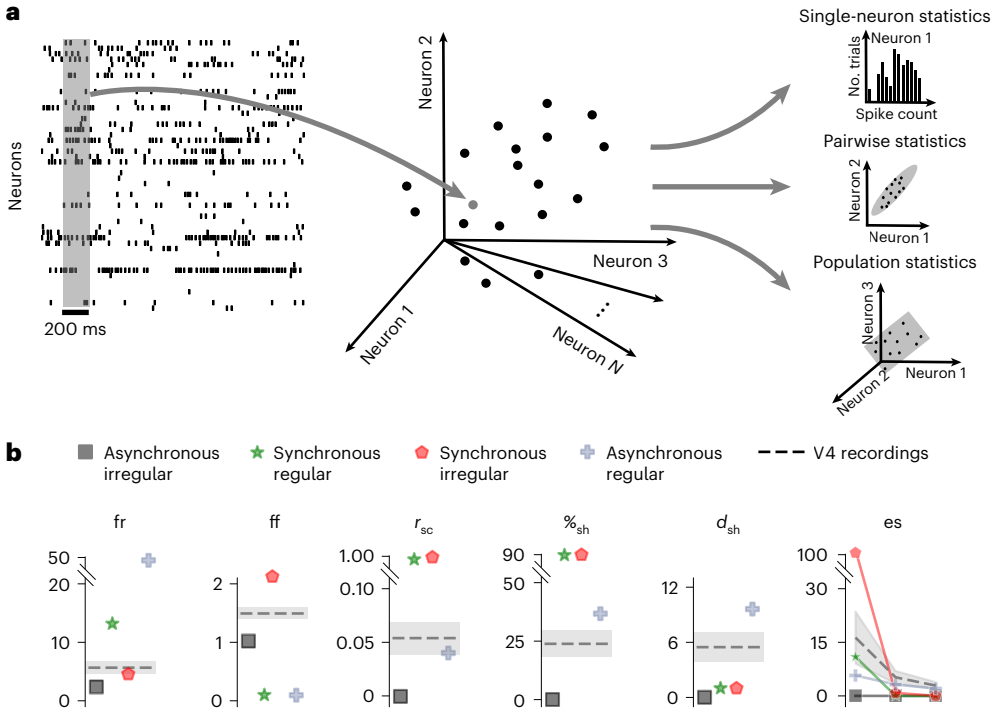

Fig. 2 | Activity statistics for comparing the activity of a spiking network model to neuronal recordings. a, The three types of activity statistics based on single neurons (for example, fr and ff), pairs of neurons (for example, spike count correlation) and a population of neurons (for example, percent shared variance, number of dimensions and eigenspectrum of shared variance). The units of fr are spikes per second, those of ff are spike count and those of the eigenspectrum of shared variance are (spike count)^2. All other activity statistics are unitless. These activity statistics are all based on spike counts within a 200 ms spike count bin (left), which can be represented in a population activity space (center). Each dot represents the activity across the neuronal population within a given time window. **b**, The activity statistics based on population recordings in macaque visual area V4 (dashed lines) were challenging to reproduce by the four parameter regimes of a CBN (colored symbols, cf. Fig. 1a, mean across five network instantiations of network connectivity graphs and initial membrane potentials corresponding to the same network parameter set). None of the four activity regimes accurately reproduced the activity statistics of the V4 population recordings (dashed lines). The V4 activity statistics are shown as the mean ± 1 s.d. across 19 recording sessions (Methods). The spike counts for V4 were computed using a fixed 200 ms time window preceding the onset of each stimulus presentation. All activity statistics were based on randomly subsampling 50 neurons from each CBN or V4 dataset.

captures the activity variability of individual neurons across time[33]. For the pairwise statistic, we computed the spike count correlation between pairs of neurons (r_{sc}, Fig. 2b), which is widely used to measure the correlated variability among neurons[34]. Both single-neuron and pairwise statistics have been widely used to customize network models to neuronal recordings[9,13,17,20,26,35].

There can also be structure in the population-wide variability that is not apparent when considering only single-neuron and pairwise statistics[32]. Previous studies have used population-wide activity statistics to compare network models with recorded activity[26,31,36,37]. Thus we also considered population activity statistics based on dimensionality reduction[38]. Specifically, we used factor analysis (FA), which is the most basic dimensionality reduction method that separates the variance that is shared among neurons from the variance that is independent to each neuron. We computed the following three population activity statistics based on FA (Fig. 2b, Methods and Supplementary Fig. 1): (1) the percent shared variance ($\%_{sh}$) is the fraction of a neuron's activity variance that is shared with one or more of the other neurons in the recorded population. This value is first computed per neuron, then averaged across neurons. A high $\%_{sh}$ indicates that the population of neurons strongly covary, whereas zero $\%_{sh}$ indicates that neurons are independent of each other. While $\%_{sh}$ is related to r_{sc}, it is not identical and captures a different aspect of population activity[32]. For example, a zero r_{sc} might correspond to a case where some pairs of neurons have positive correlations and some have negative correlations. In this case, the $\%_{sh}$ will be nonzero, reflecting the presence of shared activity among neurons. (2) We measured the dimensionality as the number of dimensions needed to explain the shared variance among neurons (d_{sh}). If the neurons all simply increase and decrease their activity together, d_{sh} will equal one. If the neurons covary in more complex ways, d_{sh} will be greater than one. (3) The eigenspectrum (es) of the shared covariance matrix measures the relative dominance of the dimensions identified above. It may be that the first dimension explains far more shared variance than the other dimensions (in which case the eigenspectrum would have a sharp dropoff) or that all dimensions explain a similar amount of shared variance (in which case the eigenspectrum would be flat).

Manual customization of SNNs to neuronal activity

It is challenging to manually customize a SNN to neuronal recordings because it can be difficult to intuit the activity of the network resulting from changes to its parameters. To get around this difficulty, one might ask whether any of the four CBN activity regimes shown in Fig. 1a capture the key aspects of neuronal recordings (Fig. 2a, left). We thus computed the single-neuron, pairwise and population activity statistics of the spike trains shown in Fig. 1a for each of the four activity regimes (Fig. 2b, colored shapes). We compared them with the activity statistics computed from neuronal recordings in macaque visual area V4 during a spatial attention task (Fig. 2b, dashed lines). Recordings were performed using a 100 electrode Utah array in a 10 × 10 configuration. We analyzed activity from 19 experimental sessions. We took spike counts within a 200 ms window preceding the onset of each stimulus presentation. To compare the activity statistics from network models and neuronal recordings on equal footing, we subsampled the recordings from each session down to 50 neurons and 700 spike count bins.

https://doi.org/10.1038/s43588-024-00688-3

We found that none of the four previously identified CBN activity regimes recapitulated all of the activity statistics of the V4 neurons. Thus, we needed an automatic method to search the parameter space to determine whether there existed a parameter set whose activity better resembled neuronal recordings.

Customizing SNNs using BO

The central contribution of this work is an automatic framework, SNOPS (Fig. 3), to address this need. SNOPS iteratively updates the parameters of the SNN so that the activity statistics of the model-generated activity (Fig. 3a) better match those of the recorded neuronal activity (Fig. 3b). To quantify how well matched are the two sets of activity statistics, we define a cost function (Fig. 3c) as a linear combination of the squared difference between the two sets of activity statistics (Methods).

Assessing how adjusting any model parameter influences the cost requires generating spiking activity from the network model. Therefore, the cost function cannot be expressed in a closed form with respect to the model parameters and cannot be optimized using gradient methods. Meanwhile, exhaustive search methods, such as random search, may yield excessive running time because they are computationally demanding for the network model to generate spikes. Instead, we need a guided way of searching the parameter space. BO is a natural choice to optimize a cost function whose evaluation depends on a time-consuming simulation[39]. It automatically proposes the next model parameter set to evaluate based on the cost of the previously evaluated parameter sets. The key idea is that more similar model parameter sets should correspond to more similar costs. In our method, this relationship is described by a Gaussian process (GP), a common choice for BO due to its closed-form expressions. The algorithm uses the GP to propose model parameters whose predicted cost is low (that is, parameter sets that may be better than those already considered) and whose uncertainty about the predicted cost is high (to sample from unvisited areas of the parameter space). This defines an exploitation–exploration tradeoff.

The GP (Fig. 3d, solid line) approximates the cost function $c(\boldsymbol{\theta})$ (Fig. 3d, dashed red line), which is a priori unknown, using all evaluations of the cost function from previous iterations and the current iteration (Fig. 3d, dots). BO will then construct an acquisition function (Fig. 3e) based on the GP-predicted cost and its uncertainty at each setting of the model parameters $\boldsymbol{\theta}$ (Methods). The parameters $\boldsymbol{\theta}^\star$ that maximize the acquisition function are selected for the next iteration, and the entire process restarts (that is, the new parameter set $\boldsymbol{\theta}^\star$ is used to simulate spike trains from the SNN, whose activity statistics are then computed and so on). With more iterations, BO will probably sample parameter sets with lower cost until a stopping criterion has been reached (see example in Supplementary Fig. 2). To further accelerate the customization procedure, we introduced two computational innovations in SNOPS: (1) running a short simulation to assess whether a parameter set is likely to yield valid spike trains (feasibility constraint; Methods) and (2) dynamically increasing the number of simulations to reduce the variance of the estimated cost (intensification; Methods).

Recovering activity statistics in simulation using SNOPS

To validate SNOPS, we first generated activity from a CBN and computed its activity statistics (Fig. 4a). These served as the target activity statistics in the customization procedure, in place of activity statistics computed from neuronal recordings. We then used SNOPS to customize a separate CBN to these activity statistics. In this case, there is no model mismatch. Thus, there exists a CBN parameter set that reproduces the target activity statistics exactly.

For comparison, we repeated the customization task with two other optimization algorithms applicable to large-scale SNNs that do not have a closed-form cost function: random search and its accelerated variant. Random search proceeds by sampling parameter sets

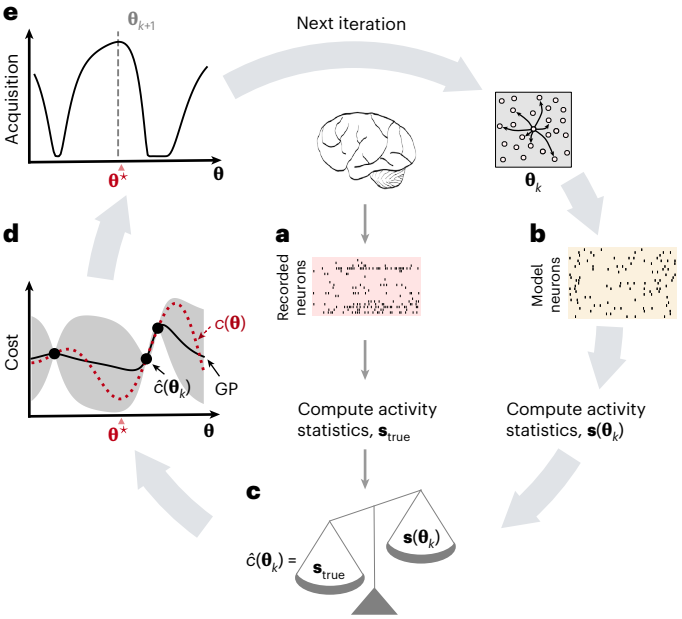

Fig. 3 | Customizing a spiking network model using BO with GPs. The BO algorithm attempts to find a parameter set $\boldsymbol{\theta}$ for a spiking network model such that its activity statistics match those of neuronal recordings. **a**, Spike trains are recorded from the brain and their activity statistics, $\mathbf{s}_{\text{true}}$, are computed. This step is performed only once, since the same recorded activity is used for comparison on all iterations. **b**, On the k-th iteration, spike trains are generated from the network model using parameter set $\boldsymbol{\theta}_k$, proposed by the previous iteration. **c**, The activity statistics of the spike trains generated from the network model, $\mathbf{s}(\boldsymbol{\theta}_k)$, are computed. The cost for $\boldsymbol{\theta}_k$ depends on how far each of those activity statistics is from the corresponding activity statistics of the neuronal recordings, $\mathbf{s}_{\text{true}}$. The intensification procedure occurs between **b** and **c**. The feasibility of $\boldsymbol{\theta}_k$ is determined here using a short simulation. **d**, A GP (solid line) is used to approximate the true, unknown cost function, $c(\boldsymbol{\theta}_k)$ (dashed red line). We seek to find the minimum of this true, unknown cost function (denoted by $\boldsymbol{\theta}^\star$). Each iteration of the BO provides one evaluation of the cost at a particular setting of the model parameters (black dots). The cost at the current iteration is labeled $\hat{c}(\boldsymbol{\theta}_k)$ and the other black dots represent the costs evaluated during previous iterations. The GP provides an uncertainty of our estimate of the cost function (gray shading). For illustrative purposes, we show here a single model parameter being optimized, whereas our algorithm typically optimizes multiple model parameters simultaneously. A separate GP is used to approximate the feasibility of $\boldsymbol{\theta}_k$ (not shown). **e**, An acquisition function is defined based on the two GPs in **d** to determine the next parameter set, $\boldsymbol{\theta}_{k+1}$, to evaluate. The acquisition function implements an exploration–exploitation tradeoff, where areas of low predicted cost and high uncertainty are desirable.

from the search region uniformly at random. This method is similar to the exhaustive search approach in previous literature[14] and provides a benchmark for performance comparisons. The accelerated random search incorporates two computational innovations that we introduced in SNOPS (feasibility constraint and intensification; Methods). Therefore, when going from random search to its accelerated variant, the only difference is incorporating the two innovations in SNOPS. Going from accelerated random search to SNOPS, the only difference is replacing random search with BO. This arrangement enables us to systematically qualify the benefits of the two key features of SNOPS: BO and the two innovations.

SNOPS (Fig. 4b and Supplementary Fig. 3, blue) outperformed accelerated random search (Fig. 4b and Supplementary Fig. 3, red), indicating that BO achieves a lower cost than random search after the same amount of computer running time. Accelerated random search outperformed random search (Fig. 4b and Supplementary Fig. 3, green), indicating that the two innovations of SNOPS are beneficial. Furthermore, all three methods yielded a CBN whose activity statistics

https://doi.org/10.1038/s43588-024-00688-3

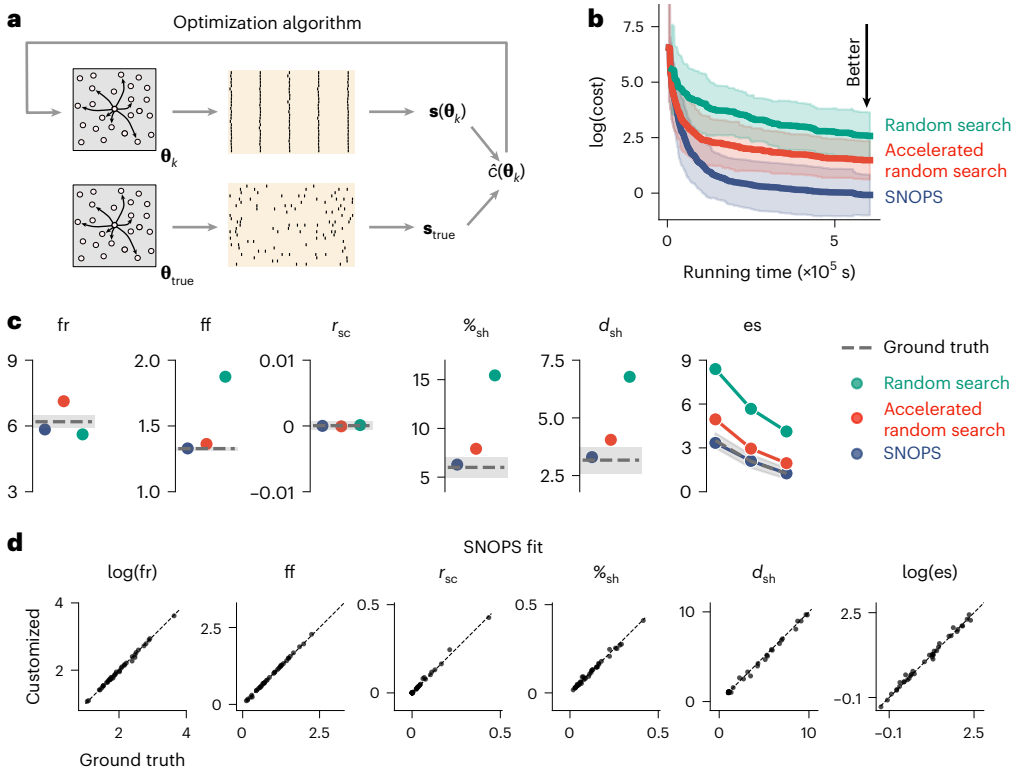

Fig. 4 | Accurate customization of a CBN model to simulated spike trains using SNOPS. a, A CBN was used to generate spike trains with randomly chosen parameter sets $\boldsymbol{\theta}_{\text{true}}$ (Methods). SNOPS (or other optimization algorithms) was then used to customize the parameters, $\boldsymbol{\theta}_k$, of a separate CBN to match the 'ground truth' activity statistics, $\mathbf{s}_{\text{true}}$, of the generated spike trains. **b**, For a given amount of computer running time (Methods), SNOPS (blue) finds parameters with lower cost than accelerated random search (red) and random search (green). The vertical axis represents the lowest log (cost) up to the given running time and hence decreases monotonically. The solid lines and shading represent the mean ± 1 s.d. across 40 customization runs. Note that 10^5 s equals 1.2 days or 27.8 hours. **c**, For a representative customization run, SNOPS (blue) identified model parameters whose activity statistics were closer to the ground truth (dashed lines) than accelerated random search (red) and random search (green). The error bars on the ground truth represent one s.d. across five network instantiations corresponding to the same ground truth parameter set. The circles represent the mean across five network instantiations corresponding to the network parameter set identified by each optimization algorithm. **d**, Across all 40 customization runs, SNOPS accurately reproduced the ground truth activity statistics (all points lie near the diagonal). Each dot represents the results from one SNOPS customization run to a randomly generated ground truth dataset. For visual clarity, only the first (that is, most dominant) mode of es is plotted in the rightmost image.

better match the target activity statistics with increasing running time, as expected (Fig. 4b and Supplementary Fig. 3). Another related method, Sequential Neural Posterior Estimator (SNPE)[28], returns a distribution of parameter sets and requires generating a large number of SNN simulations upfront. We compare SNOPS with SNPE (Supplementary Figs. 4 and 12 and Discussion) and a genetic algorithm (Supplementary Fig. 3). SNOPS outperforms both methods in customizing network models to a variety of datasets.

The cost (Fig. 4b, vertical axis) is a summary of how accurately the activity statistics of the customized CBN match the target activity statistics. To further understand the difference in performance between these methods, we then compared the individual activity statistics returned by each method with their target values. Consistent with Fig. 4b, SNOPS was better able to match the target activity statistics than the other methods (Fig. 4c). Across all 40 customization runs, SNOPS successfully identified CBNs whose activity statistics closely matched the target activity statistics (Fig. 4d, all the dots are located near the diagonal). In sum, SNOPS customizes a spiking network model to neuronal activity more quickly and accurately than the other methods.

Customizing SNNs to V4 and PFC population recordings

We next present a case study of using SNOPS to customize SNNs to neuronal population recordings in macaque monkeys (from Utah arrays implanted in visual cortical area V4 and in PFC).

In Fig. 2b, we demonstrated that none of the four CBN activity regimes from Fig. 1a recapitulated the V4 dataset (mean cost ± s.d., asynchronous irregular: 13.56 ± 0.12; synchronous regular: 1,823.67 ± 190.06; synchronous irregular: 1,489.71 ± 124.25; asynchronous regular: 361.44 ± 9.30). Here, we used SNOPS to automatically customize a CBN to the same V4 datasets and obtained a substantially lower cost (2.71 ± 24, $P < 1 \times 10^{-5}$ for each of the four comparisons, one-sided t-test). In other words, SNOPS reproduced the activity statistics more accurately than any of the four previously identified activity regimes (Fig. 5a, compare with Fig. 2b).

Despite this improvement, there were activity statistics that were not accurately reproduced. Specifically, the r_{sc} for the customized CBN was substantially smaller than that of the V4 datasets (mean ± s.d., 0.00085 ± 0.0011 versus 0.054 ± 0.015, $P < 1 \times 10^{-4}$, one-sided t-test) (Fig. 5a). To verify the reliability of this disagreement, we reran SNOPS with different initializations and obtained the same disagreement in r_{sc} (Supplementary Fig. 5). This indicates that the disagreement in r_{sc} was probably not due to a limitation of SNOPS.

We did not observe such a disagreement in r_{sc} when we customized the CBN to the simulated activity generated by CBNs across a wide range of model parameters (Fig. 4d, third graph). This led us to hypothesize that the CBN model framework is not flexible enough to capture the full complexity of the V4 datasets, as measured by the six activity statistics. We thus explored a more powerful SNN model with the goal of more accurately capturing the properties of spiking activity in the V4 datasets.

https://doi.org/10.1038/s43588-024-00688-3

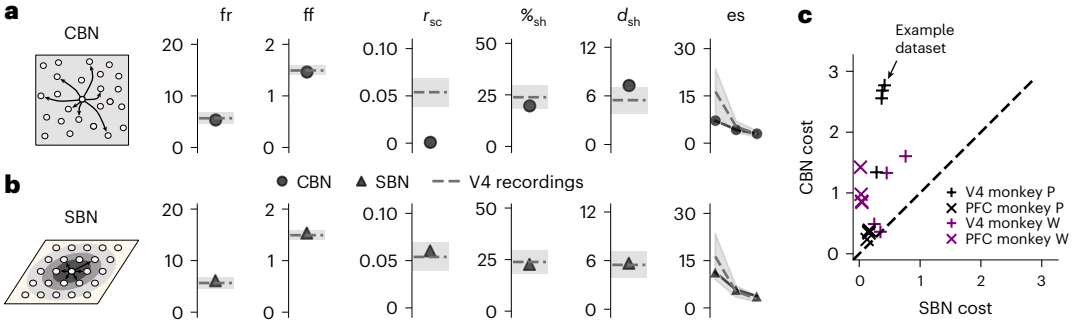

Fig. 5 | Reproducing activity statistics of macaque V4 and PFC recordings with the CBN and SBN. a, Left: stylized representation of the CBN. Right: activity statistics of the CBN (circles, mean across five network instantiations corresponding to the same identified parameter set) after being customized using SNOPS to the same V4 dataset as in Fig. 2b. The dashed line and shading represent the mean ± 1 s.d. across 19 sessions. **b**, Left: stylized representation of the SBN. The SBN is different from the CBN in that the connection probability depends on the distance between neurons. Right: activity statistics of the SBN (triangles, mean across five network instantiations corresponding to the same identified parameter set) after being customized using SNOPS to the same V4 datasets as in **a**. **c**, The SBN more accurately reproduced activity statistics than the CBN across 16 datasets, comprising four task conditions with recordings in two brain areas (V4 and PFC) in each of the two monkeys. The arrow indicates the example V4 dataset shown in **a** and **b**.

The SBN, an extension of the CBN, has been recently proposed as a SNN framework capable of producing a wider range of population statistics[11,26]. Neurons in a SBN are organized over a two-dimensional spatial lattice and have connection probabilities that depend on the distance between neuron pairs. The SBN captures well-known spatial effects of activity statistics, such as the dependence of r_{sc} on distance[26,40]. This introduces three additional model parameters, for a total of 11 parameters (Methods). This is in contrast to a CBN model, which lacks spatial connectivity and has connection probabilities that are the same for all neuron pairs. SBNs have been heuristically shown to produce activity that resembles the V4 population activity[26]. However, this claim has not been quantitatively verified. We next used SNOPS to systematically explore the capacity for SBNs to capture a wider range of population activity.

We first verified that SNOPS can accurately customize a SBN to simulated activity (Supplementary Figs. 5b and 6), mirroring our results with the CBN (Fig. 4d and Supplementary Fig. 5a). We then customized a SBN to V4 population activity and found that a SBN is able to more accurately reproduce the activity statistics of the V4 population recordings than a CBN (mean cost ± s.d., 0.26 ± 0.10 versus 2.71 ± 0.24, $P < 1 \times 10^{-5}$, one-sided t-test; Fig. 5b). To test whether the benefit of the SBN over CBN is data specific, we customized both models to each of the 16 'datasets', comprising four task conditions with recordings in two brain areas (V4 and PFC) from two monkeys (Methods). Across these datasets, the SBN consistently outperformed the CBN in reproducing the activity statistics of the neuronal recordings (Fig. 5c and Supplementary Fig. 7). We can also customize a SBN with task-dependent parameters to neuronal recordings from multiple task conditions (Supplementary Fig. 8).

Revealing limits of network model flexibility using tradeoffs in activity statistics

To understand why the SBN outperforms the CBN, we customized each SNN to each activity statistic individually rather than all six activity statistics together. We found that the CBN was able to accurately reproduce each activity statistic individually, including r_{sc} (Fig. 6a). This suggests that the reason why the CBN is unable to reproduce all six activity statistics simultaneously is due to tradeoffs between different statistics: adjusting the model parameters to better reproduce one statistic can affect how accurately another statistic is reproduced.

We thus defined a tradeoff cost, which measures whether more accurately reproducing one activity statistic leads to less accurately reproducing another activity statistic. For example, a model might be able to accurately reproduce the $\%_{sh}$, but at the expense of making r_{sc} too low. In this case, there is a nonzero tradeoff cost, indicated by a combined cost of customizing the two statistics simultaneously that is greater than customizing them individually (Fig. 6b). Note that the tradeoff cost is distinct from the overall cost, in that an accurate model with a low overall cost might still have a nonzero tradeoff cost for particular pairs of statistics.

We used the tradeoff cost to understand why the SBN can more accurately reproduce activity statistics of neuronal recordings than the CBN (cf. Fig. 5). We found that the CBN suffers from a tradeoff cost between r_{sc} and ff, as well as between r_{sc} and the population statistics (Fig. 6c, top). By contrast, the SBN has a small tradeoff cost for all pairs of statistics (Fig. 6c, bottom). This is due to the flexibility afforded to the SBN by the extra parameters that control the spatial scales of connection probabilities that the CBN lacks (Methods). Note that the average number of incoming connections is the same for the CBN and SBN. The primary distinction between the two models is that the SBN tends to have more connections between nearby neurons (controlled by the connection widths), whereas in the CBN connection probability does not depend on distance. A consequence of this flexibility is that the SBN needs to be appropriately constrained during the customization process. For example, if we customize a SBN using only single-neuron and pairwise statistics, the population statistics of the SBN are not accurately reproduced (Supplementary Fig. 10). This demonstrates the value of including population statistics in the customization process, especially for more flexible models such as the SBN.

Tradeoffs can also occur among more than two statistics. To investigate this, we systematically increased the number of statistics included in the cost function. The average cost of the customized statistics increases as more statistics are included (Fig. 6d). This illustrates how customizing a model to simultaneously reproduce more statistics imposes more constraints on the model customization process. For the CBN, there is already a marked increase in cost when going from one statistic to two statistics included in the cost function (Fig. 6d, top). In particular, there is a high cost of customizing r_{sc} and ff simultaneously (highlighted dot), consistent with Fig. 6c (top). By contrast, the average cost for the SBN remains low for even up to all six simultaneously customized statistics (Fig. 6d, bottom). Hence the pairwise tradeoff cost we show in Fig. 6c is sufficient for the comparison of model flexibility. In the multidimensional space of activity statistics, these tradeoffs form a boundary of combinations of statistics that each model is able to reproduce (Supplementary Fig. 11): the tighter boundary of the CBN than the SBN confirms its higher tradeoffs, as shown in Fig. 6c.

Discussion

A major application of SNOPS is to facilitate the development of more flexible models and thereby further our scientific understanding of

https://doi.org/10.1038/s43588-024-00688-3

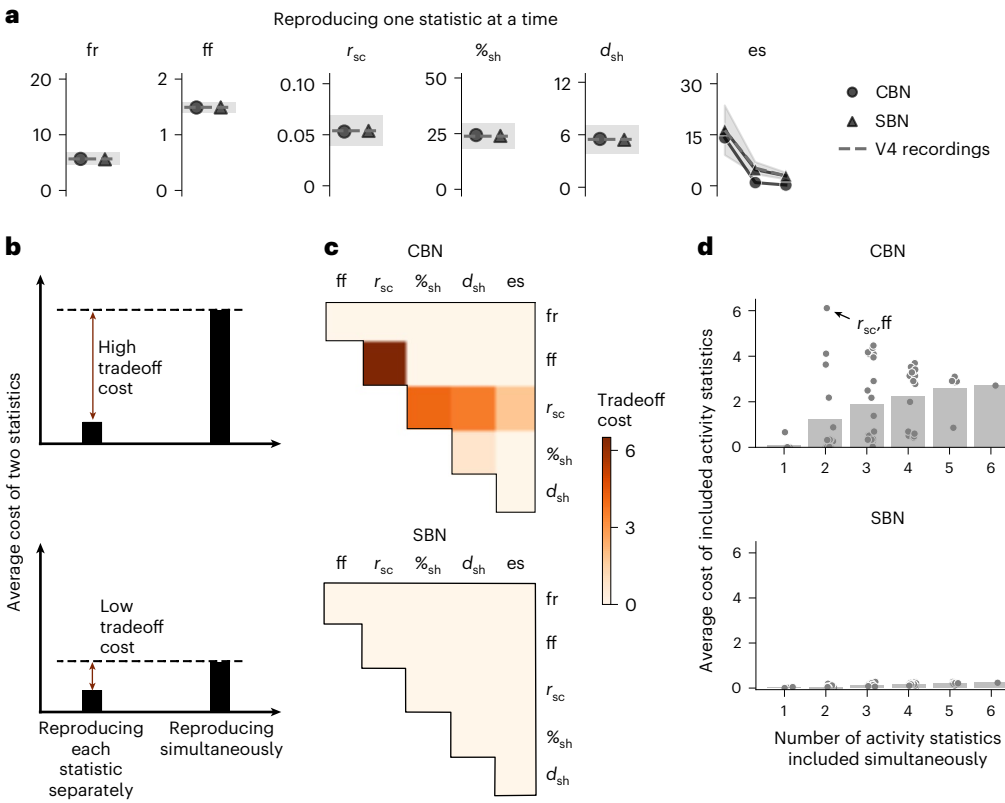

Fig. 6 | Revealing the inflexibility of CBN relative to SBN with tradeoff cost. **a**, Activity statistics of the CBN (circles, mean across five network instantiations corresponding to the same identified parameter set) and SBN (triangles, mean across five network instantiations corresponding to the same identified parameter set) after being customized using SNOPS to one V4 activity statistic (dashed line) at a time. The same V4 dataset as Fig. 2b was used. **b**, A high tradeoff cost represents the case where customizing the network to reproduce two activity statistics simultaneously yields a higher average cost of the two statistics than customizing each statistic individually (top). By contrast, a low tradeoff cost represents the case where the cost of customizing two activity statistics simultaneously yields a similar cost to customizing each statistic individually (bottom). **c**, Tradeoff costs between pairs of statistics for the CBN (top) and SBN (bottom) on the same V4 dataset as Fig. 2b. We observed similar effects when customizing these models to PFC recordings (Supplementary Fig. 9). **d**, Customizing the CBN and SBN to different numbers of activity statistics included in the cost function simultaneously, on the same V4 dataset as Fig. 2b. Each dot represents one particular subset of activity statistics (for example, highlighted dot indicates the average cost of r_{sc} and ff when including only those two activity statistics in the cost function). The cost of each dot was computed over five network instantiations corresponding to the same identified parameter set of that dot. Each bar indicates the average cost across all subsets of the corresponding number of activity statistics.

brain function. This is achieved in the following two ways. First, if certain activity statistics are not accurately reproduced during manual customization, it is unclear whether one needs to continue to manually tune the model parameters in hopes of reproducing all activity statistics or to consider a new class of models (for example, by introducing spatial connectivity). SNOPS performs a guided search of the high-dimensional parameter space, thereby providing greater confidence about when a new class of models needs to be considered. Second, the automatic optimization algorithm in SNOPS enables repeated customization of a model with different subsets of activity statistics, facilitating a more complete understanding of a model's limitations. For example, customization of a CBN to neuronal recordings might suggest that the CBN is incapable of reproducing experimentally observed r_{sc} values (cf. Fig. 5a). In this case, one could be misled to invest time in modifying the network model specifically so that it can reproduce the experimentally observed r_{sc}. Using SNOPS, we found that customizing r_{sc}, in itself was not problematic. Instead, it was the tradeoff between r_{sc} and other activity statistics that limited the CBN (cf. Fig. 6c). Such insights will not only profoundly influence plans for making the model more flexible, but also shed light on how different network architectures (for example, CBN versus SBN) lead to different model flexibility.

We emphasize the benefit of incorporating population statistics to compare the activity of network models and neuronal recordings. Indeed, population statistics have been widely used to study decision-making[17], working memory[41], attention[26,42], motor control[43], learning[44] and more. Including population statistics yields a more faithful reproduction of the neuronal activity compared with including only single-neuron or pairwise statistics (cf. Supplementary Fig. 10).

There are several key considerations when selecting which method to use for customizing a network model: properties of the cost function, the simulation time of the model, the number of customized models desired for the scientific goal and the available computational resources. The first consideration is whether the cost function has a closed-form expression with respect to model parameters. If so, evaluating the cost can be fast and one can utilize algorithms such as FORCE[13] to customize the network model. If the cost function is also differentiable with respect to the model parameters, one can customize a network model using methods that utilize the gradient, such as backpropagation[18] and emergent property inference (EPI)[45]. These approaches are computationally fast and scalable to a large number of parameters. By contrast, the cost function of large-scale SNNs typically has no closed-form expression with respect to the model parameters and hence falls outside the scope of the aforementioned methods. In such cases, three types of algorithms can be used: (1) evolutionary algorithms, which are biologically inspired, have been applied to the Hodgkin–Huxley model[46–48]; (2) SNPE[28], a method based on deep neural networks and Bayesian inference, has also been applied to customize these models to find a distribution of the parameter sets whose activity

statistics mimic those of the recordings; finally, (3) BO, as we propose here in SNOPS, can be used to customize large-scale SNNs.

The second consideration is the simulation time of the network model. SNPE requires generating a large number of simulations to train the deep neural networks. This is feasible for models with short simulation time, such as Hodgkin–Huxley and stomatogastric ganglion models. Once trained, SNPE can be used to customize the network model repeatedly to a large number of datasets without the need to run additional simulations (Supplementary Fig. 12). Large-scale SNNs, however, have a long simulation time due to the large number of neurons in the network, which poses a challenge for simulation-intensive methods such as SNPE. In such settings, optimization-based methods, such as BO, are preferred as they minimize the cost function iteratively without needing to pregenerate a large number of simulations (Supplementary Fig. 4).

The third consideration is the number of customized models desired for the scientific goal. Most studies to date customize a single model to the neuronal recordings (for example, refs. 6,7,9,11,14,26). In this case, SNOPS is preferred because it finds a single customized model more quickly and probably better reproduces the recorded activity than SNPE (cf. Supplementary Fig. 4). SNPE can also be used in this case by selecting the mode of the posterior distribution. However, the running time would be substantially greater for SNPE because it aims to capture the entire distribution of the parameters, which requires substantially more network simulations. Interestingly, there may exist different combinations of parameters that lead to the same network activity[49]. One may seek to interrogate how different parameters compensate each other to produce the same activity[28,45]. Although SNOPS can be used in this scenario, it requires repeated customization runs with different starting points to obtain multiple solutions, which is computationally demanding. In this case, deep neural networks, such as EPI and SNPE, are preferred because they return a distribution of multiple parameter sets that lead to the same activity properties. However, doing so requires either differentiability (EPI) or the ability to generate a large number of simulations quickly (SNPE).

The fourth consideration is the available computing resources. The large number of simulations needed for training the deep neural network in SNPE can be parallelized, thereby reducing the overall running time. By contrast, parallelizing SNOPS can be more challenging due to its iterative nature. It is still possible to leverage multiple cores to concurrently evaluate different parameter sets during BO (Methods), thereby accelerating the SNOPS customization.

Our approach is modular and each component of SNOPS can be tailored to the scientific goals of the user. First, we can task SNOPS with replicating additional features of neuronal activity (for example, timescales of activity) by incorporating the appropriate activity statistics (for example, autocorrelation[50,51]) in the cost function. Second, we can consider replacing GPs with a more scalable model, such as neural processes[52], to accelerate the customization process. Third, we can compare the distribution of the activity statistics, instead of their mean, in the cost function using distributional metrics such as the Wasserstein distance or Kullback–Leibler divergence. These and other extensions of the SNOPS framework may be necessary when applying SNOPS to other brain areas and/or when using other types of network models (for example, refs. 53,54).

Advancements in neuronal recording technologies are enabling measurements of brain activity at unprecedented scale. Large-scale models and large-scale neuronal recordings are closely related: large-scale models provide a systematic and mechanistic understanding of large-scale neuronal recordings, whereas large-scale neuronal recordings can further expose limitations of large-scale models. SNOPS can be used to accelerate this cycle and facilitate the synergy between model-based (mathematical) approaches and empirical measurements of brain activity to further our understanding of the brain.

Methods

Components for customizing a network model

Here, we list the components one needs for customizing a network model to neuronal population activity. Each of these components is described in detail in the sections below.

- Neuronal recordings: neuronal activity recorded from a population of neurons (or generated from a network model). In this study, the neuronal activity is in the form of spike trains either recorded experimentally or generated by a spiking network model.
- Network model with unknown parameters: a mathematical model to be customized to the neuronal recordings. In this study, we use a CBN[7] and a SBN[26].
- Activity statistics: types of activity statistics that are used to measure how similar the activity produced by the network model is to the neuronal recordings. The user can define their own activity statistics depending on their needs. In this study, we use mean fr, ff, spike count correlation (r_{sc}), percent shared variance ($\%_{sh}$), dimensionality (d_{sh}) and the eigenspectrum of the shared variance (es).
- Cost function: a function that takes as input the activity statistics of the network model and those of the neuronal recordings, and outputs a scalar that summarizes how different are the two sets of activity statistics. In this study, we use a weighted sum of squared differences.
- Optimization algorithm: an algorithm for adjusting the parameters of the network model so that its activity resembles the neuronal recordings (that is, to minimize the cost). In SNOPS, we use BO. Users can also incorporate other optimization algorithms, such as random search and evolutionary algorithms.

Spiking network models

Model details. Since both CBNs and SBNs are composed of the same single-neuron model, we will present both network models together. Each network has one feedforward layer and one recurrent layer. The feedforward layer contains $N_F = 2{,}500$ excitatory neurons emitting spikes according to independent Poisson processes with a uniform rate of 10 spikes per second. As in ref. 26, here, we use homogeneous Poisson processes as the feedforward input to primarily attribute the r_{sc} observed in the SBN to its spatial connectivity, rather than correlations inherited from the inputs. Note that other approaches, such as using external inputs with specified input correlations, have been explored to induce neuronal correlations within network models[25,55]. There are $N_e = 2{,}500$ excitatory neurons and $N_i = 625$ inhibitory neurons in the recurrent layer. Note that this number is smaller than our past work[26] (where $N_F = 2{,}500$, $N_e = 40{,}000$ and $N_i = 10{,}000$); this was done to reduce the simulation time while maintaining similarly rich network activity (Supplementary Fig. 13). The membrane potential of a neuron j in population $\alpha \in \{e, i\}$, V_j^α, in the recurrent layer obeys exponential integrate-and-fire membrane dynamics[56]

$$C_m \frac{dV_j^\alpha}{dt} = -g_L\left(V_j^\alpha - E_L\right) + g_L \Delta_T e^{\left(V_j^\alpha - V_T\right)/\Delta_T} + I_j^\alpha(t). \quad (1)$$

The passive membrane properties are given by the leak conductance g_L, the leak reversal potential E_L and the membrane capacitance C_m. The second term on the right hand side of equation (1) models the excitable membrane nonlinearity that causes an explosive spike onset for V_j^α above the soft threshold V_T (Δ_T gives the sensitivity of spike onset). Each time $V_j^\alpha(t)$ exceeds a voltage threshold V_{th}, the neuron spikes and the membrane potential are held for a refractory period τ_{ref}, then reset to a fixed value V_{re}. Here, $V_{re} < V_T < V_{th}$. The neuron model parameters are set to $\tau_m = C_m/g_L = 15$ ms, $E_L = -60$ mV, $V_T = -50$ mV, $V_{th} = -10$ mV, $\Delta_T = 2$ mV, $V_{re} = -65$ mV and $\tau_{ref} = 1.5$ ms. For inhibitory

neurons, $\tau_m = 10$ ms, $\Delta_T = 0.5$ mV and $\tau_{ref} = 0.5$ ms. Similar model formulations have been used in past studies[11,26,57], and we used the same parameters and constants as in our previous work[26] unless otherwise specified. There are also model parameters that are not fully explored in previous literature and need to be determined as the goal of the customization. We call them 'free parameters', to be introduced below.

Let the spike train from neuron k in population $\alpha \in \{e, i, F\}$ be $y_k^\alpha(t) = \sum_n \delta(t - t_{kn}^\alpha)$, where $\delta(t-s)$ is the Dirac delta function centered at time $t = s$. The total synaptic current to a neuron j in population α is

$$\frac{I_j^\alpha(t)}{C_m} = \sum_{k=1}^{N_F} \frac{J_{jk}^{\alpha F}}{\sqrt{N}} y_k^F * \eta^F(t) + \sum_{\beta=e,i} \sum_{k=1}^{N_\beta} \frac{J_{jk}^{\alpha\beta}}{\sqrt{N}} y_k^\beta * \eta^\beta(t) + \mu^\alpha, \quad (2)$$

where $N = N_e + N_i = 3{,}125$, asterisk denotes convolution and $\mu^\alpha = 0$ mV ms^{-1} is the static external input. The synaptic connection strength $J_{jk}^{\alpha\beta}$ is equal to $J^{\alpha\beta}$ if neuron k in population β connects to neuron j in population α, otherwise it is set to zero ($\alpha \in \{e, i\}$ and $\beta \in \{e, i, F\}$). There are then six total synaptic connection strengths: J^{ei}, J^{ii}, J^{ie}, J^{ee}, J^{eF} and J^{iF}, which are free parameters. The synaptic kernel from population β, $\eta^\beta(t)$, is given by

$$\eta^\beta(t) = \frac{H(t)}{\tau^{\beta d} - \tau^{\beta r}} \left(e^{-t/\tau^{\beta d}} - e^{-t/\tau^{\beta r}} \right), \quad (3)$$

where $H(t) = 0$ for $t < 0$ and $H(t) = 1$ for $t \geq 0$. The time constants $\tau^{er} = \tau^{ir} = \tau^{Fr} = 1$ ms, $\tau^{Fd} = 5$ ms and τ^{ed} and τ^{id} are free parameters.

For the CBN, the connection probability from a presynaptic neuron in population β to a postsynaptic neuron in population α is set to a constant $\bar{p}^{\alpha\beta}$. Throughout we used $\bar{p}^{ee} = 0.15$, $\bar{p}^{ei} = 0.6$, $\bar{p}^{ie} = 0.45$, $\bar{p}^{ii} = 0.6$, $\bar{p}^{eF} = 0.1$ and $\bar{p}^{iF} = 0.05$, to enable the network to display a similar average number of connections as the model in our previous work[26].

For the SBN, neurons in the two layers are arranged uniformly on a 1 mm square grid. The probability of a connection from presynaptic neuron k in population β located at position (x_k, y_k) to postsynaptic neuron j in population α located at position (x_j, y_j) is

$$p = \bar{p}^{\alpha\beta} g(x_j - x_k; \sigma^{\alpha\beta}) g(y_j - y_k; \sigma^{\alpha\beta}), \quad (4)$$

where $g(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \sum_{k=-\infty}^{\infty} e^{-(x+k)^2/(2\sigma^2)}$ is a wrapped Gaussian distribution. The connection widths ($\sigma^{ee} = \sigma^{ie} = \sigma^e$, $\sigma^{ei} = \sigma^{ii} = \sigma^i$, $\sigma^{eF} = \sigma^{iF} = \sigma^F$) are free parameters for the SBN. Note that mathematically, the SBN is more flexible than the CBN because the latter can be considered a special case of the former: setting the three parameters that control connectivity width to infinity will turn a SBN into a CBN. For both the CBN and SBN, self-connections are allowed. There can be more than one connection between any given pair of neurons, in which case the effective connection strength between the neurons is the number of connections times $J^{\alpha\beta}$.

The summary of the free parameters common to both the CBN and SBN network models is presented in Supplementary Table 1. The SBN model has additional free parameters presented in Supplementary Table 2.

Model simulation. To simulate activity from the network, we first instantiated a network model. This involves generating a network connectivity graph based on the connection probabilities, and setting the initial membrane potential of each neuron from a uniform distribution between −65 and −50 mV, as in our previous work[26]. We will discuss the impact of the randomness induced by the realization of the connectivity graph and initial membrane potentials in the following sections. After model instantiation, the differential equations were solved by the forward Euler method using a time step of 0.05 ms for a duration of the simulation predetermined by the user. Future work may use a more efficient integration scheme in place of the Euler's method. For example, the fourth-order Runge–Kutta method with adaptive step size can be used to adjust the step size based on the estimated error of the numerical solution to speed up the simulation process. However, to take advantage of the increased accuracy from higher-order numerical integration methods, we will need to interpolate within the time step when the voltage crosses the spike threshold (that is, $V_j^\alpha(t) \geq V_{th}$) so as to identify the spike time with an accuracy that matches the order of the numerical scheme[58].

Identifying the four activity regimes for the CBN. For Figs. 1 and 2, we needed to identify CBN parameters that correspond to each of the four activity regimes introduced in ref. 30. As the network architecture in ref. 30 is different from ours (in terms of the number of neurons, choice of integrate-and-fire neuron model and so on), the parameter values in their paper are not directly applicable to our network. Thus, we randomly sampled 5,000 parameter sets from the search range of the CBN (Supplementary Table 1). We then selected parameter sets that produced each of the following four combinations of statistics: low r_{sc} and high ff (asynchronous irregular), high r_{sc} and low ff (synchronous regular), high r_{sc} and high ff (synchronous irregular) and low r_{sc} and low ff (asynchronous regular). Figure 1a shows the spike trains of 50 randomly selected neurons over a period of 200 ms for each activity regime. For the analysis in Fig. 2b, we simulated 140.5 s of spiking activity and computed the activity statistics of 50 randomly sampled neurons for each of the four activity regimes (see 'Estimating activity statistics' section).

Activity statistics

Let $X \in \mathbb{R}^{N_s \times T}$ be a matrix of spike counts taken in a fixed time window (defined below) for N_s sampled neurons (either from the neuronal recordings or SNN) and T time bins. On the basis of X, we computed the following activity statistics (illustrated in Supplementary Fig. 1):

Single-neuron statistics. We considered two commonly used single-neuron statistics: fr and ff. The fr is defined as the mean fr across all neurons and trials. Specifically, we average all elements of X and divide by the duration of the spike count window.

The ff measures the trial-to-trial variability of the activity of each neuron. For each neuron (that is, row of X), we compute its ff as the variance of the T values divided by the mean of the T values. We then average these ff values across all neurons. For reference, if the spike counts for each neuron were Poisson distributed, then ff would equal 1.

Pairwise statistic. We considered the pairwise spike count correlation (r_{sc}), commonly used to measure how pairs of neurons covary[34]. The r_{sc} was computed by first computing the Pearson correlation for each pair of neurons across the T trials, then averaging the correlation values across all $N_s(N_s - 1)$ pairs of neurons. We applied the Fisher transformation[40] when comparing r_{sc} values in the cost function because it makes the r_{sc} values more Gaussian distributed, as in previous work[40]:

$$z = \frac{1}{2} \log\left(\frac{1 + r_{sc}}{1 - r_{sc}} \right). \quad (5)$$

Population statistics. We considered three statistics that characterize population-wide covariability: the percent shared variance ($\%_{sh}$), the dimensionality of the shared variance (d_{sh}) and the eigenspectrum of the shared variance (es)[31,32]. These statistics are based on FA, the most basic dimensionality reduction method that partitions variance that is shared among neurons from the variance that is independent to each neuron. Note that principal component analysis does not distinguish between these two types of variance.

Using the spike count matrix X, we can compute the $N_s \times N_s$ covariance matrix C. For consistency, we used exactly the same spike counts

for computing the single-neuron, pairwise and population statistics (that is, we counted spikes in the same time bins). FA performs the decomposition $C \approx LL^{\mathrm{T}} + \Psi$, where $L \in \mathbb{R}^{N_s \times m}$ is the loading matrix, $\Psi \in \mathbb{R}^{N_s \times N_s}$ is a diagonal matrix containing the independent variance of each neuron and m is the number of latent dimensions. The matrix LL^{T} represents the variance shared among neurons (termed the 'shared covariance matrix') and Ψ represents the variance independent to each neuron. The FA parameters L and Ψ are estimated from the neuronal activity using the expectation-maximization algorithm. The number of latent dimensions m is determined by maximizing the fivefold cross-validated data likelihood.

On the basis of these FA parameters, we define three population statistics. The $\%_{\mathrm{sh}}$ quantifies the percentage of each neuron's variance that is shared with one or more of the other simultaneously recorded neurons. This value is then averaged across neurons. Specifically, we compute

$$\%_{\mathrm{sh}} = \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{L_{j,:} L_{j,:}^{\mathrm{T}}}{L_{j,:} L_{j,:}^{\mathrm{T}} + \Psi_j}, \tag{6}$$

where $L_{j,:}$ represents the j-th row of L, and Ψ_j represents the j-th diagonal element of Ψ. Note that $\%_{\mathrm{sh}}$ is related, but not equivalent, to r_{sc}[32].

The d_{sh} measures the complexity of the shared variance among neurons (that is, the number of population activity patterns needed to describe the shared variance). For example, if all neurons increased and decreased their activity together, d_{sh} would equal 1. In principle, we should choose $d_{sh} = m$. In practice, we first found m by maximizing the cross-validated data likelihood, as described above. Then, we chose d_{sh} as the number of dimensions needed to explain 95% of the shared variance (based on the eigenspectrum of LL^{T}). This procedure increases the reliability of the estimated dimensionality[31].

The es measures the relative dominance of the dimensions of shared variance. For example, m might equal 3, but one dimension might explain far more shared variance than the other two dimensions. Specifically, es is defined as the vector of N_s eigenvalues of LL^{T}, where the eigenvalues are ordered from largest to smallest. Only the first m eigenvalues are nonzero. We define es in this way so that two eigenspectra with different m can be directly compared.

Estimating activity statistics. To estimate the activity statistics of the SNN with a given parameter set, we first instantiated the model (which includes generating a network connectivity graph and initial membrane potentials, see 'Model simulation' section). For estimating the activity statistics in Figs. 2 and 4–6, we repeated this network instantiation procedure five times and averaged the estimated statistics over these repetitions to increase estimation reliability (see below). For a given network instantiation, we simulated the network (see 'Spiking network models' section) to obtain 140.5 s of spiking activity. We then removed the first 500 ms of the spike train to ensure the statistics are computed on the spike trains when the network has reached a stable state, similar to Huang et al.[26]. We also excluded neurons whose fr is less than 0.5 spikes per second for stable estimation of variance-based statistics (see 'Feasibility constraints' section). We then binned the remaining 140 s into 700 bins, each of duration 200 ms. We used 140 s of activity with 700 bins because empirically such a number of bins is sufficient for a stable estimation of the aforementioned activity statistics while still keeping the simulation time reasonably low (average of 373 s, see 'SNOPS running time' section) for our spiking network model with 5,625 neurons.

For the spike trains corresponding to a given network instantiation, we computed their activity statistics using 50 randomly sampled excitatory neurons in the recurrent layer (similar to Huang et al.[26]). We sampled 50 neurons to compare model output spike trains directly with that of recorded neuronal population activity, where the number of recorded neurons is typically around 50. We repeatedly sampled 50 neurons without replacement from the network model ten times. The activity statistics were then computed for each sampled population and averaged across the ten samplings. This reduces the sampling variance in the estimation of the activity statistics. If there are five network instantiations, we further averaged the activity statistics over these instantiations.

For the neuronal recordings, we first excluded neurons with frs less than 0.5 spikes per second. We then randomly sampled 50 neurons and 700 trials without replacement for each recording session and condition, where each trial corresponds to a single stimulus presentation (see 'Neuronal recordings' section). Within each trial, we took spike counts in a 200 ms bin preceding stimulus onset. Hence, the activity statistics for the network model and neuronal recordings are both computed using 50 neurons and 700 trials to ensure consistency for the comparisons.

Neuronal recordings

Experiments were approved by the Institutional Animal Care and Use Committee of the University of Pittsburgh and were performed in accordance with the United States National Research Council's Guide for the Care and Use of Laboratory Animals. We reanalyzed data from experiments reported in previous studies[42,59]. In brief, we trained two rhesus macaque monkeys (monkeys P and W) to perform a spatial attention task. At the beginning of each task trial, the animal first fixated on a central dot for 300–500 ms. Gabor stimuli were presented, one on each side of fixation, for 400 ms. One of the two stimulus locations was block cued to change its orientation with 90% probability. After the end of the stimulus presentation, a blank interstimulus period of 300–500 ms followed. The described sequence repeated and on each presentation, there was a fixed probability of one of the Gabor stimuli changing orientation at each presentation (that is, a flat hazard function). The task of the animal was to detect a change in orientation of one of the two stimuli and make a saccade to the stimulus that changed. Thus, the animal would benefit from maintaining constant attention to the cued location throughout the task trial.

Two 100 electrode Utah arrays (Blackrock Microsystems, one in V4 and one in PFC) were used to record neuronal activity in V4 and PFC simultaneously during the spatial attention task. There were two cue conditions (attention directed to the aggregate V4 receptive field or to the other hemifield) and two stimulus orientations (45° and 135° with the hemifields always containing orthogonal orientations), leading to four unique task conditions with different frs and population statistics. For each condition, we included only recording sessions with at least 50 neurons whose fr is greater than 0.5 spikes per second each and at least 700 stimulus presentations for accurate estimation of the activity statistics (see 'Estimating activity statistics' section). This yielded 10 sessions for V4 of monkey W, 20 sessions for PFC of monkey W, 19 sessions for V4 of monkey P and 19 sessions for PFC of monkey P. More sessions were excluded for V4 than PFC for monkey W because many V4 neurons in monkey W had frs less than 0.5 spikes per second. We included both successful and failed trials because we looked at the 200 ms bin preceding stimulus onset which was largely unaffected by the eventual trial result. Since on each trial the monkey saw multiple flashes of the stimulus, we took a 200 ms spike count bin immediately preceding each flash (that is, stimulus onset), leading to multiple spike count bins per task trial. We customized the network models to each of the four conditions separately (see 'Customizing CBN and SBN to neuronal recordings' section).

Cost function

We measured the discrepancy between the SNN-generated activity and neuronal recordings using a cost function. Specifically, our cost function is a weighted linear combination of the normalized distance of each activity statistic from its target value. Let S be the set of statistics

included in the cost function. For example, $S = \{\text{fr}, \text{ff}, r_{sc}, \%_{sh}, d_{sh}, \text{es}\}$ indicates that all six activity statistics are used for customizing the SNN. Let s_j^{true} and $s_j(\boldsymbol{\theta})$ denote the jth activity statistic of the neuronal recordings (that is, the target value) and that of a network model under parameter set $\boldsymbol{\theta}$, respectively, where $j \in S$. The cost function is defined as:

$$c_S(\boldsymbol{\theta}) = \frac{1}{\sum_{j\in S} w_j} \sum_{j\in S} w_j \frac{d\left(s_j^{\text{true}}, s_j(\boldsymbol{\theta})\right)}{v_j^{\text{true}}}, \tag{7}$$

where $d(\cdot, \cdot)$ is a distance function. In this work, $d(s_j^{\text{true}}, s_j(\boldsymbol{\theta})) = (s_j^{\text{true}} - s_j(\boldsymbol{\theta}))^2$. The weight, $w_j \in (0, 1)$, indicates the relative importance of each statistic and is predefined by the user. If a weight is zero, the corresponding activity statistic is not used during the customization procedure. In this work, we set $w_j = 1$ for all j because we wanted to weigh each statistic equally. The terms s_j^{true} and v_j^{true} are the mean and variance of the j-th activity statistic across simulations or recording sessions (defined below). The variance term serves to downweight a statistic if its variance is large, indicating the estimation is unreliable. For the eigenspectrum of the shared covariance matrix (es), $d(\cdot, \cdot)$ is defined as the sum of squared differences of the corresponding elements in the eigenspectra. The variance term for es is then computed across sessions or recording sessions using this scalar value.

For the network model (Fig. 4), s_j^{true} and v_j^{true} are the mean and variance, respectively, of the corresponding statistic over five network instantiations with randomly generated graphs and initial membrane potentials corresponding to the same ground truth parameter set. For neuronal recordings (Figs. 5 and 6), s_j^{true} and v_j^{true} are the mean and variance, respectively, across multiple recording sessions from the same monkey and experimental condition. In the sections below, we will refer to $c_S(\boldsymbol{\theta})$ as simply $c(\boldsymbol{\theta})$, where S will be clear from the context.

Optimization algorithm

Problem setup. The goal of the optimization algorithm is to find a parameter set $\boldsymbol{\theta} \in \mathbb{R}^d$ in the search region Θ that minimizes $c(\boldsymbol{\theta})$ as

$$\min_{\boldsymbol{\theta}\in\Theta} c(\boldsymbol{\theta}). \tag{8}$$

In practice, $c(\boldsymbol{\theta})$ does not have a closed-form expression in terms of the model parameters and cannot be optimized using gradient-based methods. This is because, for a large-scale spiking network model, $c(\boldsymbol{\theta})$ depends on several activity statistics, which in turn depend on the computationally demanding numerical simulation of the SNN (see 'SNOPS running time' section). Hence, for a given $\boldsymbol{\theta}$, we cannot compute $c(\boldsymbol{\theta})$ directly as a function of $\boldsymbol{\theta}$. Instead, we simulate the network to obtain an estimate of $c(\boldsymbol{\theta})$, denoted $\hat{c}(\boldsymbol{\theta})$. The estimation error is $c(\boldsymbol{\theta}) - \hat{c}(\boldsymbol{\theta})$ and arises from several sources. First, for a given $\boldsymbol{\theta}$, network connectivity graphs are randomly generated. This is because the network connectivity graph is not a parameter of the network model, but is instead drawn from probability distributions specified by the parameters $\boldsymbol{\theta}$. Second, for a given graph, initial membrane potentials of each neuron are drawn randomly to ensure diversity of membrane potentials in the neuronal population, as in our previous work[26]. Third, the network has multiple layers, where the neurons in the first layer (the feedforward layer) emit spikes according to independent Poisson processes. Hence the spike trains from the first layer will differ under the same connectivity graph and initial membrane potentials.

In the following sections, we will introduce two optimization algorithms (BO and random search) to minimize $c(\boldsymbol{\theta})$, and two innovations (feasibility constraints and intensification) to accelerate optimization. Both innovations can be incorporated into BO or random search. We term BO with both innovations 'SNOPS' (Fig. 4, blue), random search with both innovations 'accelerated random search' (Fig. 4, red) and random search without innovations 'random search' (Fig. 4, green).

Random search. An intuitive approach to minimizing $c(\boldsymbol{\theta})$ without a closed-form expression is random search. Random search is commonly used as a benchmark in optimization and has been shown to have similar performance to more advanced algorithms in many optimization tasks[60]. At each iteration, the algorithm randomly samples a parameter set uniformly from the search region Θ and evaluates its cost. The algorithm terminates after a user-defined number of iterations, K, has been reached (Algorithm 1).

To reduce the variance of $\hat{c}(\boldsymbol{\theta})$, we repeatedly simulate spike trains with randomly generated graphs and initial membrane potentials using the same parameter set for R repetitions ($R = 5$ in this work). For each repetition, we evaluate the cost, then average across the repetitions (Algorithm 1, the inner loop). Note that we will improve this variance-reduction method using one of the innovations (that is, intensification) to be introduced later.

Algorithm 1: Random search for SNN customization.

Input: search region Θ; max number of iterations K; number of repeated simulations R.
Initialization: previously sampled parameter sets $\hat{\Theta} = \{\}$ and their costs $\hat{C} = \{\}$.
for $k \leftarrow 1{:}K$ **do**
  $\boldsymbol{\theta}_k \sim \text{uniform}(\Theta)$.
  for $r \leftarrow 1{:}R$ **do**
    Evaluate $\hat{c}(\boldsymbol{\theta}_k)_r$ by simulating spike trains with a randomly generated graph and initial membrane potentials.
  end for
  $\hat{c}(\boldsymbol{\theta}_k) = \frac{1}{R}\sum_{r=1}^{R} \hat{c}(\boldsymbol{\theta}_k)_r$.
  $\hat{\Theta} \leftarrow \hat{\Theta} \cup \{\boldsymbol{\theta}_k\}$.
  $\hat{C} \leftarrow \hat{C} \cup \{\hat{c}(\boldsymbol{\theta}_k)\}$.
end for
return $\boldsymbol{\theta}^\star \leftarrow \underset{\boldsymbol{\theta}\in\hat{\Theta}}{\text{argmin}}\, \hat{C}$.

Algorithm 2: BO for SNN customization.

Input: search region Θ; max number of iterations K; number of repeated simulations R.
Initialization: sample 50 initial parameter sets, either uniformly at random or according to a prior distribution, to obtain Θ. For each element $\boldsymbol{\theta} \in \hat{\Theta}$, estimate its cost $\hat{c}(\boldsymbol{\theta}) = \frac{1}{R}\sum_{r=1}^{R} \hat{c}(\boldsymbol{\theta})_r$, where the cost estimate is averaged over R repeated simulations. Then, define $\hat{C} = \{\hat{c}(\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}$.
for $k \leftarrow 1{:}K$ **do**
  Fit $\mathcal{GP}$ to $(\hat{\Theta}, \hat{C})$.
  Compute $\boldsymbol{\theta}_k \leftarrow \underset{\boldsymbol{\theta}\in\Theta}{\text{argmax}}\, a(\boldsymbol{\theta})$, where $a(\boldsymbol{\theta})$ is the acquisition function in equation (10).
  for $r \leftarrow 1{:}R$ **do**
    Evaluate $\hat{c}(\boldsymbol{\theta}_k)_r$ by simulating spike trains with a randomly generated graph and initial membrane potentials.
  end for
  $\hat{c}(\boldsymbol{\theta}_k) = \frac{1}{R}\sum_{r=1}^{R} \hat{c}(\boldsymbol{\theta}_k)_r$.
  $\hat{\Theta} \leftarrow \hat{\Theta} \cup \{\boldsymbol{\theta}_k\}$.
  $\hat{C} \leftarrow \hat{C} \cup \{\hat{c}(\boldsymbol{\theta}_k)\}$.
end for
return $\boldsymbol{\theta}^\star \leftarrow \underset{\boldsymbol{\theta}\in\hat{\Theta}}{\text{argmin}}\, \hat{C}$.

BO. Random search samples parameter sets independently at each iteration and is not guided by the previously sampled parameter sets. To accelerate the algorithm, we turn to BO. BO utilizes previous evaluations of the cost to guide the parameter search in a way that promotes both exploration and exploitation. BO has been demonstrated to optimize cost functions with fewer iterations than random search in various optimization tasks[39].

BO involves two major components: (1) a GP model to approximate the cost function and (2) an acquisition function to determine the

parameter set to sample at the next iteration. The full algorithm of BO involves iteratively updating the GP model and proposing the next parameter set using the acquisition function. This is outlined in Algorithm 2.

First, BO uses a GP to approximate $c(\boldsymbol{\theta})$. If two sets of parameters, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are similar, we expect the corresponding costs $c(\boldsymbol{\theta}_1)$ and $c(\boldsymbol{\theta}_2)$ to also be similar. To capture this intuition, BO approximates the cost function as a smooth function of the model parameters using a GP. The GP will allow us to predict $c(\boldsymbol{\theta})$ (posterior mean of the GP) and our uncertainty about the value of $c(\boldsymbol{\theta})$ (posterior variance of the GP) for a candidate $\boldsymbol{\theta}$ without performing the computationally demanding evaluation of $c(\boldsymbol{\theta})$ explicitly. Specifically, we write

$$\hat{c}(\boldsymbol{\theta}) \sim \mathrm{GP}(\mu(\boldsymbol{\theta}), \kappa(\cdot,\cdot)), \tag{9}$$

where $\mu(\boldsymbol{\theta}) : \Theta \mapsto \mathbb{R}$ is the mean function of the GP and is set as a constant, 0, without loss of generality. The covariance function, $\kappa(\cdot,\cdot) : \Theta \times \Theta \mapsto \mathbb{R}$, is a positive definite kernel function defined on any two points in the search region, Θ. We use the automatic relevance determination Matérn 5/2 kernel. The Matérn 5/2 kernel is commonly used in BO because it allows for possible nonsmoothness of a cost function. Its automatic relevance determination variant fits a different length scale for each of the d elements of $\boldsymbol{\theta}$, as determined by data. Note that incorporating prior knowledge of parameter interactions into the kernel can be beneficial (Supplementary Fig. 14). We use the MATLAB function fitgpr for fitting a GP model to the sampled parameter sets and their associated costs. In practice, the GP is fit to the cost estimated by averaging the estimate over $R = 5$ network instantiations of the same parameter set to reduce variance (as in random search). The uncertainty in the estimated costs is captured by the signal standard deviation in the Matérn 5/2 kernel, which is fit along with other hyperparameters in the kernel. We also log-transformed the cost values when fitting the GP to mitigate the effect of extreme cost values. At the beginning of the optimization, a set of initial parameter sets, Θ, is sampled uniformly to fit the GP since we assume no prior knowledge about the location of the optimal parameter set (Algorithm 2). One may sample Θ according to a prior distribution other than the uniform distribution to guide the initial optimization process if one has such knowledge. We sampled 50 initial parameter sets for Θ, although this number can be varied based on user need. The larger this number, the better the initial GP estimate of $c(\boldsymbol{\theta})$ will be, but the longer the initialization process will take.

Second, BO uses an acquisition function based on the posterior mean and variance of the GP in equation (9) to decide the next parameter set to evaluate. BO selects the parameter set at which the acquisition function value is maximized. This corresponds to a combination of low posterior cost (exploitation, where the cost is predicted to be low) and high posterior variance (exploration, where we have not sampled many parameter sets).

Let $\hat{\mu}(\boldsymbol{\theta})$ denote the posterior mean and $\hat{\sigma}(\boldsymbol{\theta})$ denote the posterior standard deviation of the GP at $\boldsymbol{\theta}$. Let f_- be the minimum of $\hat{c}(\theta)$ over the sampled parameter sets so far. We use the expected improvement[39] as the acquisition function

$$a(\boldsymbol{\theta}) = (f_- - \hat{\mu}(\boldsymbol{\theta}))\,\Phi\left(\frac{f_- - \hat{\mu}(\boldsymbol{\theta})}{\hat{\sigma}(\boldsymbol{\theta})}\right) + \hat{\sigma}(\boldsymbol{\theta})\phi\left(\frac{f_- - \hat{\mu}(\boldsymbol{\theta})}{\hat{\sigma}(\boldsymbol{\theta})}\right), \tag{10}$$

where Φ and ϕ are the normal cumulative distribution function and probability distribution function, respectively. Equation (10) is derived based on the goal of preferring $\boldsymbol{\theta}$ whose posterior mean, $\hat{\mu}(\boldsymbol{\theta})$, is as small as possible compared with f_- (for the complete derivation, see Brochu et al.[39]). The first term of the equation represents exploitation: as $\hat{\mu}(\boldsymbol{\theta})$ becomes smaller, this term will dominate because $f_- - \hat{\mu}(\boldsymbol{\theta})$ will increase and Φ will approach 1, while ϕ in the second term will approach zero. The second term represents exploration: as $\hat{\sigma}(\boldsymbol{\theta})$ becomes larger, this term will dominate because the first term will approach 0.5 while the second term will increase as ϕ goes toward its peak.

The acquisition function, $a(\boldsymbol{\theta})$, also does not have an analytical form with respect to $\boldsymbol{\theta}$ because $\hat{\mu}(\boldsymbol{\theta})$ and $\hat{\sigma}(\boldsymbol{\theta})$ have a nonstraightforward dependence on $\boldsymbol{\theta}$. However, $\hat{\mu}(\boldsymbol{\theta})$ and $\hat{\sigma}(\boldsymbol{\theta})$ are fast to compute using fitgpr in MATLAB (typically less than a microsecond for one evaluation). Hence we evaluate $a(\boldsymbol{\theta})$ on a large number of randomly sampled $\boldsymbol{\theta}$ to quickly maximize $a(\boldsymbol{\theta})$ (as in the bayesopt function in MATLAB). In particular, we first evaluate $a(\boldsymbol{\theta})$ on 100,000 randomly sampled parameter sets. We then select ten parameter sets with the largest $a(\boldsymbol{\theta})$. We run fminsearchbnd to search locally around each of these ten parameter sets to refine the solution. The final maximizer of $a(\boldsymbol{\theta})$ is the maximizer from these ten local searches.

In Algorithm 2, the algorithm stops once the maximum number of iterations, K, has been reached. Depending on the specific use case, one can specify other stopping criteria, such as when a maximum amount of customization time has been reached, the cost function is no longer decreasing, or the cost is below a user-defined threshold.

Feasibility constraints. Our first innovation seeks to accelerate the optimization process using feasibility constraints. A parameter set, $\boldsymbol{\theta}$, is labeled 'infeasible' if, for a particular connectivity graph and initial membrane potentials (a single iteration in the inner loop in Algorithms 1 and 2), it leads to neuronal population activity generated from the network model that falls into either of the following two categories.

First, $\boldsymbol{\theta}$ may lead to extreme frs. Low frs are undesirable because the resulting spike count matrices are mostly zeros. This can lead to unstable estimates of the variance-based activity statistics (for example, ff, r_{sc} and population statistics). High frs are biologically unrealistic (typically <10 spikes per second for V4 and PFC recordings; Supplementary Fig. 5). We set the low fr threshold as <0.5 spikes per second and the high fr threshold as >60 spikes per second (mean fr across all neurons and time).

Second, $\boldsymbol{\theta}$ may lead to unstable solutions. The network activity may take a period of time to reach a stable state, defined by when the mean fr across the neuronal population converges. As noted above, a standard preprocessing step is to remove the first 500 ms of the network-generated spike trains (the period when the network has not yet stabilized)[26]. However, some parameter sets may lead to networks that take more time to stabilize or may never reach the stable state (for example, switching between multiple stable states). These cases need to be excluded from the customization process because they represent unstable solutions and are not typically used to compare with recorded neuronal activity. Specifically, to determine if a given $\boldsymbol{\theta}$ leads to an unstable solution, we first run change point detection (using the findchangepts function in MATLAB)[61] on the time course of the population-averaged mean fr after removing the first 500 ms. We then deem $\boldsymbol{\theta}$ to be infeasible if the mean fr (across neurons and time) before the change point and that after the change point exceeds a threshold. The threshold is computed as three standard deviations of the mean fr (across neurons and time) after the change point.

To speed up the customization process, we wish to rule out infeasible parameter sets with minimal computation. We propose to use a 'freeze–thaw' method[62] by first running a short simulation to generate 10 s of spike trains to estimate the feasibility of a parameter set and only proceed to the full simulation (140 s, see 'Estimating activity statistics' section) if the parameter set is feasible. We use 10 s for the short simulation because the two constraints only depend on the fr, and empirically the estimation of the fr tends to stabilize within 10 s. However, estimating population statistics, for example, $\%_{sh}$, requires substantially more simulation time, hence a full simulation is still needed to compute all activity statistics.

For random search, feasibility constraints can be incorporated in a straightforward manner: for each sampled parameter set, if feasible, the algorithm will run the full simulation of 140 s to compute the cost. If the sampled parameter set is infeasible, it will simply proceed to the

next sampled parameter set. Note that feasibility is evaluated at each of the R repetitions in the inner loop (Algorithm 1). The inner loop aborts as soon as the parameter set is deemed infeasible.

For BO, we incorporated the feasibility constraint into the optimization problem[63]

$$\min_{\boldsymbol{\theta}\in\Theta} c(\boldsymbol{\theta}) \text{ such that } g(\boldsymbol{\theta}) = 1, \tag{11}$$

where $g(\boldsymbol{\theta}) = 1$ if $\boldsymbol{\theta}$ is feasible and $g(\boldsymbol{\theta}) = 0$ otherwise.

To incorporate this constraint, two parts of BO will change. First, in addition to the GP that approximates the cost function (the GP in equation (9)), there is a separate GP that represents the feasibility function, $g(\boldsymbol{\theta})$, which is fit to the sampled parameter sets and their feasibility values (which are binary). Second, the acquisition function will now incorporate the GP for the feasibility function. Let GP_c represent the GP on $c(\boldsymbol{\theta})$ as in equation (9) and GP_g represent the GP on $g(\boldsymbol{\theta})$. Let $\hat{\mu}_c(\boldsymbol{\theta}), \hat{\sigma}_c(\boldsymbol{\theta})$ denote the posterior mean and s.d., respectively, of GP_c. Similar to equation (10), f_- is the minimum of $\hat{\mu}_c(\boldsymbol{\theta})$ over the parameter sets evaluated so far. The expected improvement (that is, acquisition) function for the constrained Bayesian optimization becomes[63]

$$a(\boldsymbol{\theta}) = \Phi\left(\frac{\hat{\mu}_g(\boldsymbol{\theta})-0.5}{\hat{\sigma}_g(\boldsymbol{\theta})}\right) \times \left((f_- - \hat{\mu}_c(\boldsymbol{\theta}))\,\Phi\left(\frac{f_- - \hat{\mu}_c(\boldsymbol{\theta})}{\hat{\sigma}_c(\boldsymbol{\theta})}\right) + \hat{\sigma}_c(\boldsymbol{\theta})\phi\left(\frac{f_- - \hat{\mu}_c(\boldsymbol{\theta})}{\hat{\sigma}_c(\boldsymbol{\theta})}\right)\right). \tag{12}$$

This differs from equation (10) in the term $\Phi\left(\frac{\hat{\mu}_g(\boldsymbol{\theta})-0.5}{\hat{\sigma}_g(\boldsymbol{\theta})}\right)$, which yields a larger acquisition value if the feasibility posterior mean is high. Note that even though GP_g is fit to binary values ($g(\boldsymbol{\theta})$ is either 0 or 1), its posterior mean, $\hat{\mu}_g(\boldsymbol{\theta})$, is continuous valued. Furthermore, in this term, $\hat{\mu}_g(\boldsymbol{\theta})$ is referenced to 0.5 to ensure symmetry.

It is also possible to replace the short simulation for evaluating the feasibility of a parameter set with analytical calculations. For instance, for networks with a large number of neurons, mean-field theory can be used to predict the mean fr directly from the network parameters[11]. In such networks, linear response theory can also be used to predict the ff and r_{sc} (ref. 11). These approaches can speed up the SNOPS customization by eliminating the need for these network simulations.

Intensification. The second innovation seeks to improve the accuracy of estimating the cost with less time. The estimation error arises from several sources, described in 'Problem setup'. A high estimation error may result in a final parameter set returned by the algorithm with a low cost in a single evaluation, but with a higher cost if evaluated and averaged over multiple repetitions[64].

One possible solution is to repeatedly run simulations under the same parameter set for R repetitions, as in Algorithms 1 and 2. However, this can be computationally demanding because each repetition corresponds to a lengthy simulation. To avoid performing R repetitions for every $\boldsymbol{\theta}$ sampled, we propose to use an intensification algorithm[64]. The main idea is to only perform R repetitions of the simulation if we encounter a potentially optimal parameter set. We first define the incumbent parameter set as the parameter set whose cost, calculated by averaging over R repetitions, is the smallest over the list of sampled parameters. Similarly, the incumbent cost and standard deviation are the associated mean and standard deviation of the cost of the incumbent parameter set over the R repetitions. A sampled $\boldsymbol{\theta}$ is considered potentially optimal if its cost evaluated in the first repetition is within one standard deviation of the incumbent cost. Otherwise, we include it in the list of sampled parameters and proceed to sample the parameter set for the next iteration. The algorithm will perform evaluations for R repetitions only for the potential optimal parameter sets. If its average cost over the R repetitions is smaller than the incumbent cost, $\boldsymbol{\theta}$ becomes the incumbent parameter set. Otherwise, we still include it in the list of sampled parameters and its associated cost value is the average over the R repetitions.

We adopt this method and introduce an additional evaluation stopping criterion: the algorithm will stop performing repetitions if the standard deviation of the cost across the performed repetitions is below a specified threshold. A small standard deviation indicates the estimate of the cost of this parameter set is consistent across repetitions and needs no variance reduction. Note that at least two repetitions need to be performed to compute the standard deviation and, if this stopping criterion is met, we follow the same procedure above in determining if $\boldsymbol{\theta}$ becomes the incumbent parameter set. The additional stopping criterion further reduces the total number of simulations throughout the optimization procedure and provides additional acceleration. We set the predefined number of repetitions to $R = 5$ and the standard deviation threshold to 0.15. A larger predefined number of repetitions and a smaller standard deviation threshold will yield a smaller estimation error, at the expense of greater simulation time.

Local optima. If SNOPS is run for infinite time, it is guaranteed to return the global optimal parameter set[39]. In practice, finite running time may result in the algorithm returning a local optimum. Empirically, we verified that SNOPS reliably returned a set of activity statistics that matched the recorded neuronal activity when the optimization algorithm was initialized with different initial parameter values (Supplementary Fig. 5). This indicates that, even if the algorithm returns a local optimum, this optimum corresponds to activity statistics that match those of the recorded activity.

Tradeoff cost

We define a tradeoff cost to measure whether more accurately reproducing one activity statistic leads to less accurately reproducing another activity statistic (Fig. 6). Intuitively, customizing two statistics s_a and s_b simultaneously might incur a larger cost (of s_a and s_b) than customizing each of them individually. The gap between the cost of customizing s_a and s_b together versus individually represents how much the two statistics tradeoff with each other. Note that a similar idea has also been explored to show the tradeoff in the loss function of a model trained to perform two tasks simultaneously[65].

For two statistics s_a and s_b, let $c_{s_a}(\theta)$ and $c_{s_b}(\theta)$ represent the cost values of the optimal parameter sets when individually customizing s_a and s_b, respectively. Let $c_{s_a \cup s_b}(\boldsymbol{\theta})$ represent the cost resulting from customizing the two statistics together. The tradeoff cost between s_a and s_b is defined as:

$$\text{tradeoff}(s_a, s_b) = c_{s_a \cup s_b}(\boldsymbol{\theta}) - \frac{c_{s_a}(\boldsymbol{\theta}) + c_{s_b}(\boldsymbol{\theta})}{2}, \tag{13}$$

where the second term represents the average of $c_{s_a}(\boldsymbol{\theta})$ and $c_{s_b}(\boldsymbol{\theta})$. This average makes the second term comparable to the first term.

The tradeoff cost is guaranteed to be nonnegative because customizing both statistics simultaneously is as challenging or more challenging than customizing each of them separately. This leads to a higher cost for each statistic when the model is customized to both statistics together (first term) as compared with when the model is customized to each of them separately (second term). A tradeoff cost of zero indicates that the ability of the model to reproduce one statistic is unaffected by the incorporation of another statistic into the cost function.

Verifying SNOPS in simulation

To validate the performance of SNOPS in simulation (Fig. 4 and Supplementary Figs. 3 and 4), we customized network models to the activity generated from the same type of model. We first randomly sampled 100 parameter sets from the search region (see 'Spiking network models' section) for the CBN and SBN. We estimated the activity statistics for each sampled parameter set over five network instantiations (see 'Cost function' section). We excluded parameter sets resulting in fr smaller than 1, d_{sh} smaller than 1, and ff larger than 5 because they do not fall

within the range of the V4 and PFC activity statistics (Supplementary Fig. 11). We then randomly sampled 40 of the remaining parameter sets for both the CBN and SBN. We chose to use 40 parameter sets because they represent a diverse combination of activity statistics while having reasonable running time. We applied SNOPS, random search and accelerated random search to customize the CBN and SBN separately. We ran each optimization-based method (SNOPS, random search and its accelerated variant) for 168 h (7 days). We found the cost usually plateaus within 168 h, indicating that the running time is sufficient for SNOPS to converge (Supplementary Fig. 3).

Customizing CBN and SBN to neuronal recordings

To compare the CBN and SBN as well as to validate the performance of SNOPS on neuronal recordings (Fig. 5 and Supplementary Figs. 3 and 4), we ran SNOPS, accelerated random search, random search and SNPE on the 16 datasets comprising two monkeys (monkeys P and W), four conditions (two cues by two saccade locations) and two brain areas (V4 and PFC). As in the previous section, we set the stopping criterion for the optimization-based methods to 168 h (7 days) because we found the cost usually plateaued within 168 h (Supplementary Fig. 3).

To compare the tradeoffs of different subsets of statistics between the CBN and SBN (Fig. 6), we customized each network model to the example macaque V4 dataset in Fig. 2 with different subsets of activity statistics included in the cost function. For each customization run, we set the weight of each statistic (w_j in equation (7)) to be either 0 or 1, leading to a total of $2^6 - 1 = 63$ customization runs for each network, which accounts for all possible subsets of activity statistics. For each customization run, we ran SNOPS for 168 h.

SNOPS running time

The running times indicated in this paragraph were obtained using cluster machines with 40 Intel Xeon Gold 6230 2.10 GHz central processing unit (CPU) cores and 250 GB of random access memory. For clarity, here we refer to one customization run as the process of customizing a SNN to one dataset using SNOPS (Algorithm 2). We used one CPU core for each customization run. The overall running time for one customization run is 168 h, corresponding to 1,200 optimization iterations on average. Each optimization iteration involves the following components. First, for a selected parameter set that maximizes the acquisition function, we randomly instantiated a network connectivity graph and initial membrane potentials and generated spike trains from the spiking network with 5,625 neurons. This is the most time-consuming part of each iteration. It takes 23 s to generate 10 s of spike trains to determine feasibility (see 'Feasibility constraints' section) and 373 s to generate 140 s of spike trains. The values are the same for the CBN and SBN since they have the same number of neurons (5,625). Second, for the generated spike trains of 140 s, it takes 69 s to compute its activity statistics. Finally, it takes 32 s to select the parameter set for the next iteration, including fitting the GP for both the feasibility constraints and cost function, as well as maximizing the acquisition function. Note that for some iterations, the spike train generation and activity statistics computation may be repeated up to five times due to the intensification procedure (see 'Intensification' section).

We can utilize multiple CPU cores to implement SNOPS with parallelization. In Figs. 4–6 and Supplementary Figs. 3–10 and 13–14, we implemented a parallelization scheme to accelerate the customization process using SNOPS. The key idea is that parameter sets evaluated by one thread (that is, one CPU core) could potentially benefit other threads, allowing for the assessment of costs without necessitating additional simulations. This is enabled by the fact that, regardless of the target activity statistics in each thread, there is a common relationship between model parameters and activity statistics (that is, that specified by the network model). For example, when customizing network models to the 40 simulated datasets in Fig. 4, we ran 40 customization runs on 40 CPU cores simultaneously. We saved the model parameters and activity statistics in a log file for each iteration of each customization run. Within each thread, we routinely checked the already-evaluated model parameters and their activity statistics in the log files of the other threads running concurrently, and computed their costs based on the target activity statistics of the current thread. If there exists a parameter set from other threads that leads to a cost that is lower than any parameter set for the current thread, we would consider this parameter set potentially optimal and proceed to evaluate it. We performed this routine check every ten iterations, although it can be done more frequently. Even though each thread involved different target activity statistics, this strategy allowed the utilization of the concurrent information across threads. To ensure a fair comparison, we applied this strategy to all optimization algorithms (random search, accelerated search, BO) in this work. Moreover, in our comparison of SNOPS with SNPE (Supplementary Fig. 4), we considered the total customization time for SNOPS as the running time of each thread (168 h) times the number of threads. This ensured a fair comparison by accounting for the collective computational effort across all threads.

Reporting summary

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

Data availability

The V4 and PFC recordings we analyzed for SNN customization are available at https://doi.org/10.1184/R1/19248827 (ref. 66). Source data for Figs. 2, 4, 5 and 6 are available with this manuscript.

Code availability

MATLAB code for the SNOPS algorithm is available at https://github.com/ShenghaoWu/SpikingNetworkOptimization and on Zenodo at https://zenodo.org/records/13218535 (ref. 67). Data analysis was performed using Python 3.8.5.

References

1. Hodgkin, A. L. & Huxley, A. F. Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *J. Physiol.* **116**, 449–472 (1952).
2. Marder, E. & Bucher, D. Understanding circuit dynamics using the stomatogastric nervous system of lobsters and crabs. *Annu. Rev. Physiol.* **69**, 291–316 (2007).
3. Vogels, T. P., Rajan, K. & Abbott, L. F. Neural network dynamics. *Annu. Rev. Neurosci.* **28**, 357–376 (2005).
4. Kass, R. E. et al. Computational neuroscience: mathematical and statistical perspectives. *Annu. Rev. Stat. Appl.* **5**, 183–214 (2018).
5. Wang, Xiao-Jing Theory of the multiregional neocortex: large-scale neural dynamics and distributed cognition. *Annu. Rev. Neurosci.* **45**, 533–560 (2022).
6. Yamins, DanielL. K. & DiCarlo, J. J. Using goal-driven deep learning models to understand sensory cortex. *Nat. Neurosci.* **19**, 356–365 (2016).
7. Van Vreeswijk, C. & Sompolinsky, H. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* **274**, 1724–1726 (1996).
8. Buonomano, D. V. & Maass, W. State-dependent computations: spatiotemporal processing in cortical networks. *Nat. Rev. Neurosci.* **10**, 113–125 (2009).
9. Hennequin, G., Vogels, T. P. & Gerstner, W. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron* **82**, 1394–1406 (2014).
10. Denève, S. & Machens, C. K. Efficient codes and balanced networks. *Nat. Neurosci.* **19**, 375–382 (2016).
11. Rosenbaum, R., Smith, M. A., Kohn, A., Rubin, J. E. & Doiron, B. The spatial structure of correlated neuronal variability. *Nat. Neurosci.* **20**, 107–114 (2017).

12. DePasquale, B., Sussillo, D. & and Mark M Churchland, L. F. Abbott The centrality of population-level factors to network computation is demonstrated by a versatile approach for training spiking networks. *Neuron* **111**, 631–649 (2023).
13. Sussillo, D. & Abbott, L. F. Generating coherent patterns of activity from chaotic neural networks. *Neuron* **63**, 544–557 (2009).
14. Stringer, C. et al. Inhibitory control of correlated intrinsic variability in cortical networks. *eLife* **5**, e19695 (2016).
15. Williamson, R. C., Doiron, B., Smith, M. A. & Yu, B. M. Bridging large-scale neuronal recordings and large-scale network models using dimensionality reduction. *Curr. Opin. Neurobiol.* **55**, 40–47 (2019).
16. Sussillo, D., Churchland, M. M., Kaufman, M. T. & Shenoy, K. V. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat. Neurosci.* **18**, 1025–1033 (2015).
17. Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013).
18. LeCun, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1**, 541–551 (1989).
19. Rajan, K., Abbott, L. F. & Sompolinsky, H. Stimulus-dependent suppression of chaos in recurrent neural networks. *Phys. Rev. E* **82**, 011903–011907 (2010).
20. Litwin-Kumar, A. & Doiron, B. Slow dynamics and high variability in balanced cortical networks with clustered connections. *Nature Neurosci.* **15**, 1498–1505 (2012).
21. Deco, G. & Hugues, E. Neural network mechanisms underlying stimulus driven variability reduction. *PLoS Comput. Biol.* **8**, e1002395 (2012).
22. Hennequin, G., Ahmadian, Y., Rubin, D. B., Lengyel, M. áté & Miller, K. D. The dynamical regime of sensory cortex: stable dynamics around a single stimulus-tuned attractor account for patterns of noise variability. *Neuron* **98**, 846–860 (2018).
23. Kumar, A., Rotter, S. & Aertsen, A. Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. *Nat. Rev. Neurosci.* **11**, 615–627 (2010).
24. Snyder, A. C., Morais, M. J., Willis, C. M. & Smith, M. A. Global network influences on local functional connectivity. *Nat. Neurosci.* **18**, 736–743 (2015).
25. Doiron, B., Litwin-Kumar, A., Rosenbaum, R., Ocker, G. K. & Josić, Krešimir The mechanics of state-dependent neural correlations. *Nat. Neurosci.* **19**, 383–393 (2016).
26. Huang, C. et al. Circuit models of low-dimensional shared variability in cortical networks. *Neuron* **101**, 337–348 (2019).
27. Billeh, Y. N. et al. Systematic integration of structural and functional data into multi-scale models of mouse primary visual cortex. *Neuron* **106**, 388–403 (2020).
28. Gonçalves, P. J. et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife* **9**, e56261 (2020).
29. Shadlen, M. N. & Newsome, W. T. The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. *J. Neurosci.* **18**, 3870–3896 (1998).
30. Brunel, N. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci.* **8**, 183–208 (2000).
31. Williamson, R. C. et al. Scaling properties of dimensionality reduction for neural populations and network models. *PLoS Comput. Biol.* **12**, e1005141 (2016).
32. Umakantha, A. et al. Bridging neuronal correlations and dimensionality reduction. *Neuron* **109**, 2740–2754 (2021).
33. Churchland, M. M. et al. Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nat. Neurosci.* **13**, 369–378 (2010).
34. Cohen, M. R. & Kohn, A. Measuring and interpreting neuronal correlations. *Nat. Neurosci.* **14**, 811–819 (2011).
35. De La Rocha, J., Doiron, B., Shea-Brown, E., Josić, Krešimir & Reyes, A. Correlation between neural spike trains increases with firing rate. *Nature* **448**, 802–806 (2007).
36. Mazzucato, L., Fontanini, A. & La Camera, G. Stimuli reduce the dimensionality of cortical activity. *Front. Syst. Neurosci.* **10**, 11 (2016).
37. Recanatesi, S., Ocker, GabrielKoch, Buice, M. A. & Shea-Brown, E. Dimensionality in recurrent spiking networks: global trends in activity and local origins in connectivity. *PLoS Comput. Biol.* **15**, e1006446 (2019).
38. Cunningham, J. P. & Yu, B. M. Dimensionality reduction for large-scale neural recordings. *Nat. Neurosci.* **17**, 1500–1509 (2014).
39. Brochu, E., Cora, V. M., & De Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Preprint at https://arxiv.org/abs/1012.2599 (2010).
40. Smith, M. A. & Kohn, A. Spatial and temporal scales of neuronal correlation in primary visual cortex. *J. Neurosci.* **28**, 12591–12603 (2008).
41. Murray, J. D. et al. Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proc. Natl Acad. Sci. USA* **114**, 394–399 (2017).
42. Snyder, A. C., Yu, B. M. & Smith, M. A. Distinct population codes for attention in the absence and presence of visual stimulation. *Nat. Commun.* **9**, 1–14 (2018).
43. Churchland, M. M. et al. Neural population dynamics during reaching. *Nature* **487**, 51–56 (2012).
44. Sadtler, P. T. et al. Neural constraints on learning. *Nature* **512**, 423–426 (2014).
45. Bittner, S. R. et al. Interrogating theoretical models of neural computation with emergent property inference. *eLife* **10**, e56265 (2021).
46. Friedrich, P., Vella, M., Gulyás, A. I., Freund, T. F. & Káli, S. A flexible, interactive software tool for fitting the parameters of neuronal models. *Front. Neuroinform.* **8**, 63 (2014).
47. Carlson, K. D., Nageswaran, J. M., Dutt, N. & Krichmar, J. L. An efficient automated parameter tuning framework for spiking neural networks. *Front. Neurosci.* **8**, 10 (2014).
48. Van Geit, W. et al. Bluepyopt: leveraging open source software and cloud infrastructure to optimise model parameters in neuroscience. *Front. Neuroinform.* **10**, 17 (2016).
49. Prinz, A. A., Bucher, D. & Marder, E. Similar network activity from disparate circuit parameters. *Nat. Neurosci.* **7**, 1345–1352 (2004).
50. Murray, J. D. A hierarchy of intrinsic timescales across primate cortex. *Nat. Neurosci.* **17**, 1661–1663 (2014).
51. Runyan, C. A., Piasini, E., Panzeri, S. & Harvey, C. D. Distinct timescales of population coding across cortex. *Nature* **548**, 92–96 (2017).
52. Garnelo, M. et al. Conditional neural processes. In *International Conference on Machine Learning* 1704–1713 (PMLR, 2018).
53. Destexhe, A., Contreras, D., Sejnowski, T. J. & Steriade, M. A model of spindle rhythmicity in the isolated thalamic reticular nucleus. *J. Neurophysiol.* **72**, 803–818 (1994).
54. Burak, Y. & Fiete, I. R. Accurate path integration in continuous attractor network models of grid cells. *PLoS Comput. Biol.* **5**, e1000291 (2009).
55. Rabinowitz, N. C., Goris, R. L., Cohen, M. & Simoncelli, E. P. Attention stabilizes the shared gain of v4 populations. *eLife* **4**, e08998 (2015).
56. Fourcaud-Trocmé, N., Hansel, D., Van Vreeswijk, C. & Brunel, N. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *J. Neurosci.* **23**, 11628–11640 (2003).

57. Brette, R. & Gerstner, W. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* **94**, 3637–3642 (2005).
58. Hansel, D., Mato, Germán, Meunier, C. & Neltner, L. On numerical simulations of integrate-and-fire neural networks. *Neural Comput.* **10**, 467–483 (1998).
59. Snyder, A. C., Yu, B. M. & Smith, M. A. A stable population code for attention in prefrontal cortex leads a dynamic attention code in visual cortex. *J. Neurosci.* **41**, 9163–9176 (2021).
60. Li, L. & Talwalkar, A. in *Uncertainty in Artificial Intelligence* (eds Adams, R. P. & Gogate, V.) 367–377 (PMLR, 2020).
61. Killick, R., Fearnhead, P. & Eckley, I. A. Optimal detection of changepoints with a linear computational cost. *J. Am. Stat. Assoc.* **107**, 1590–1598 (2012).
62. Swersky, K., Snoek, J. & Adams, R. P. Freeze-thaw Bayesian optimization. Preprint at https://arxiv.org/abs/1406.3896 (2014).
63. Gelbart, M. A., Snoek, J. & Adams, R. P. Bayesian optimization with unknown constraints. In *Proc. 30th Conference on Uncertainty in Artificial Intelligence* 250-259 (AUAI Press, 2014).
64. Hutter, F., Hoos, H. H., Leyton-Brown, K. & Murphy, K. P. An experimental investigation of model-based parameter optimisation: Spo and beyond. In *Proc. 11th Annual Conference on Genetic and Evolutionary Computation* 271–278 (Association for Computing Machinery, 2009).
65. Yang, GuangyuRobert, Joglekar, M. R., Song, H. F., Newsome, W. T. & Wang, Xiao-Jing Task representations in neural networks trained to perform many cognitive tasks. *Nat. Neurosci.* **22**, 297–306 (2019).
66. Snyder, A., Johnston, R. & Smith, M. Utah array recordings from visual cortical area V4 and prefrontal cortex with simultaneous EEG. *CMU KiltHub* https://doi.org/10.1184/R1/19248827 (2024).
67. Wu, S. Spiking network optimization using population statistics: v1.0.0. *Zenodo*https://doi.org/10.5281/zenodo.13218535 (2024).

Acknowledgements

This work was supported by National Institutes of Health (NIH) R01 NS121913 (C.H.), NIH K99 EY025768 (A.C.S.), NIH R01 EB026953 (B.D., M.A.S. and B.M.Y.), NIH R01 MH118929 (B.M.Y. and M.A.S.), National Science Foundation (NSF) Integrative Strategies for Understanding Neural and Cognitive Systems (NCS) Division of Behavioral and Cognitive Sciences (BCS) 1734916/1954107 (B.M.Y. and M.A.S.), Simons Foundation NC-GB-CULM-00002794-06 (C.H.), 542967 (B.D.) and 543065 (B.M.Y.), NSF NCS Research on Learning in Formal and Informal Settings (DRL) 2124066 (B.M.Y. and M.A.S.), NIH R01 EY029250 (M.A.S.), NIH U19 NS107613 (B.D.), Vannevar Bush faculty fellowship N00014-18-1-2002 (B.D.), NSF NCS BCS 1533672 (B.M.Y.), NIH R01 HD071686 (B.M.Y.), NIH R01 NS105318 (B.M.Y.), NIH RF1 NS127107 (B.M.Y.) and NIH R01 NS129584 (B.M.Y.). This work used the Extreme Science and Engineering Discovery Environment, which is supported by National Science Foundation grant ACI-1548562.

Author contributions

S.W., C.H., M.A.S., B.D. and B.M.Y. designed the analyses. S.W. implemented SNOPS and performed all analyses. A.C.S. and M.A.S. designed and performed the experiments. S.W., C.H., M.A.S., B.D. and B.M.Y. wrote the manuscript. All authors discussed the results and commented on the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s43588-024-00688-3.

Correspondence and requests for materials should be addressed to Byron M. Yu.

Peer review information *Nature Computational Science* thanks Valentin Dragoi and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editor: Ananya Rastogi, in collaboration with the *Nature Computational Science* team.