nature computational science

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

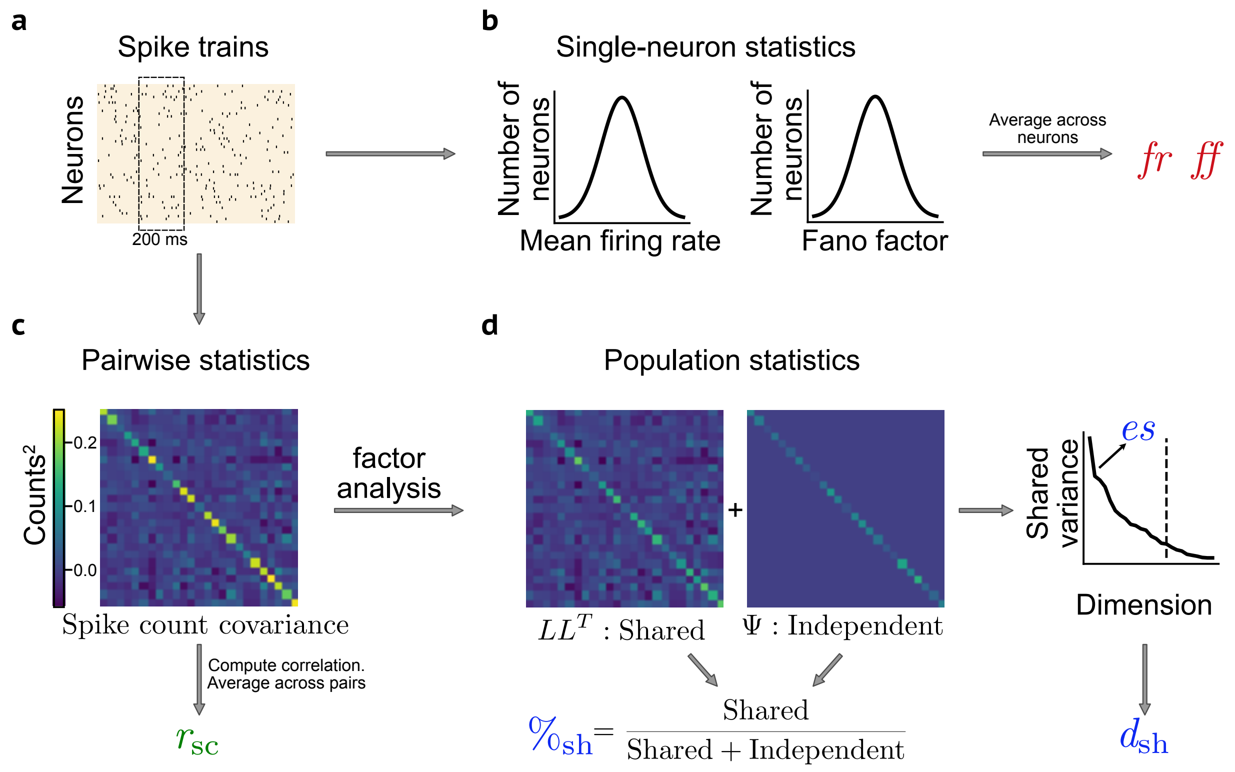
Automated customization of large-scale spiking network models to neuronal population activity

Supplementary Table 1: Free parameters for both the CBN and SBN

Parameter description	**Symbol**	**Search range**
recurrent inhibitory synaptic decay time constant	τ^{id}	1 to 25 ms
recurrent excitatory synaptic decay time constant	τ^{ed}	1 to 25 ms
recurrent I to E connection strength	J^{ei}	-150 to 0 mV
recurrent E to I connection strength	J^{ie}	0 to 150 mV
recurrent I to I connection strength	J^{ii}	-150 to 0 mV
recurrent E to E connection strength	J^{ee}	0 to 150 mV
feedforward to recurrent E connection strength	J^{eF}	0 to 150 mV
feedforward to recurrent I connection strength	J^{iF}	0 to 150 mV

Supplementary Table 2: Free parameters exclusive to the SBN

Parameter description	**Symbol**	**Search range**
recurrent inhibitory connection width	σ^{i}	0 to 0.25 mm
recurrent excitatory connection width	σ^{e}	0 to 0.25 mm
feedforward connection width	σ^{F}	0 to 0.25 mm



Supplementary Figure 1: Computing single-neuron, pairwise, and population activity statistics

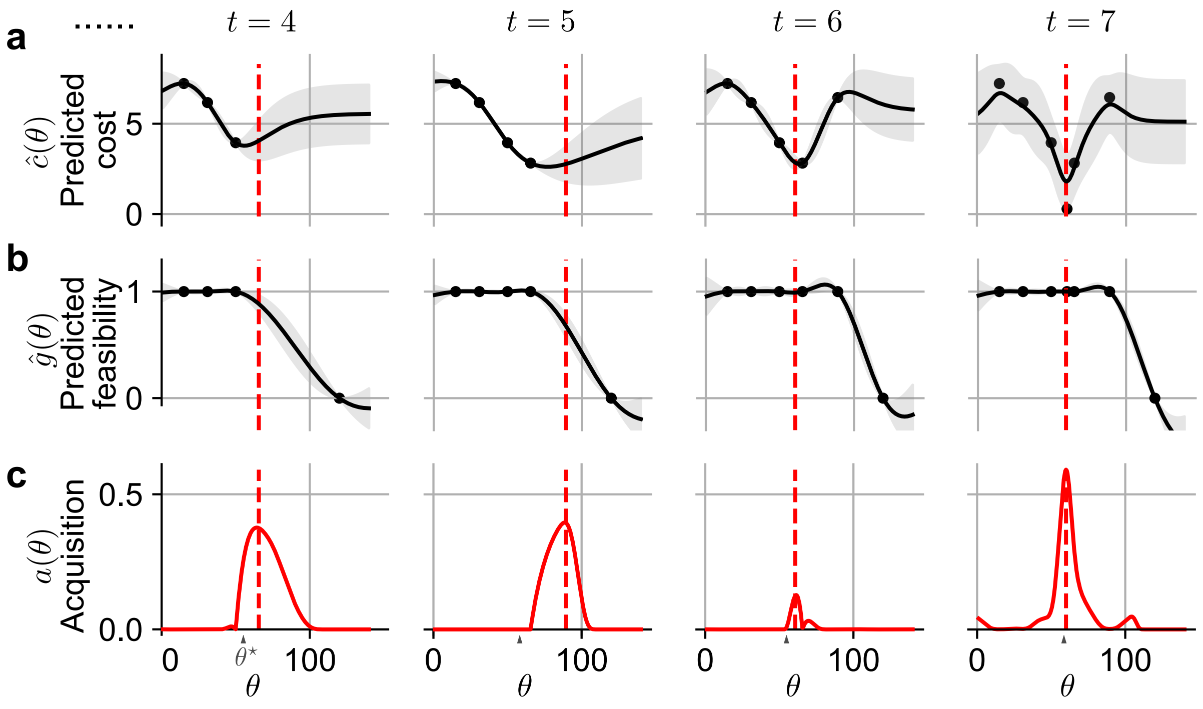
We introduced three types of activity statistics in Fig. 2. Here we illustrate how to compute these statistics from neuronal activity.

a, The spike count of each neuron is taken within a specified time window, yielding a $N_s \times 1$ spike count vector for N_s neurons. We can obtain many such spike count vectors taken in non-overlapping time windows, resulting in a $N_s \times T$ spike count matrix, with T being the number of non-overlapping time windows.

b, For the single-neuron statistics, the mean firing rate (fr) is computed by averaging the spike counts across time and neurons and dividing them by the duration of the spike count window. The Fano factor ($f\!f$) is computed as the ratio of the variance of the counts divided by the mean of the counts for each neuron, then averaged across neurons.

c, The spike count correlation (r_{sc}) between pairs of neurons is obtained by first computing the $N_s \times N_s$ covariance matrix of spike counts. The Pearson correlation is computed for each pair of neurons, then averaged across pairs.

d, For the population statistics, the spike count covariance matrix is first decomposed into a shared covariance matrix (LL^{T}) and an independent variance matrix (Ψ) using factor analysis[1]. The percent shared variance ($\%_{\mathrm{sh}}$) is the percent of each neuron's spike count variance that is shared with other neurons in the recorded population. This value is then averaged across neurons. The dimensionality (d_{sh}) is the number of latent dimensions needed to explain 95% of the shared variance. The eigenspectrum (es) comprises the amount of shared variance represented by each latent dimension. Mathematically, it is the eigenspectrum of LL^{T}.



Supplementary Figure 2: Using SNOPS to customize a SNN with one parameter

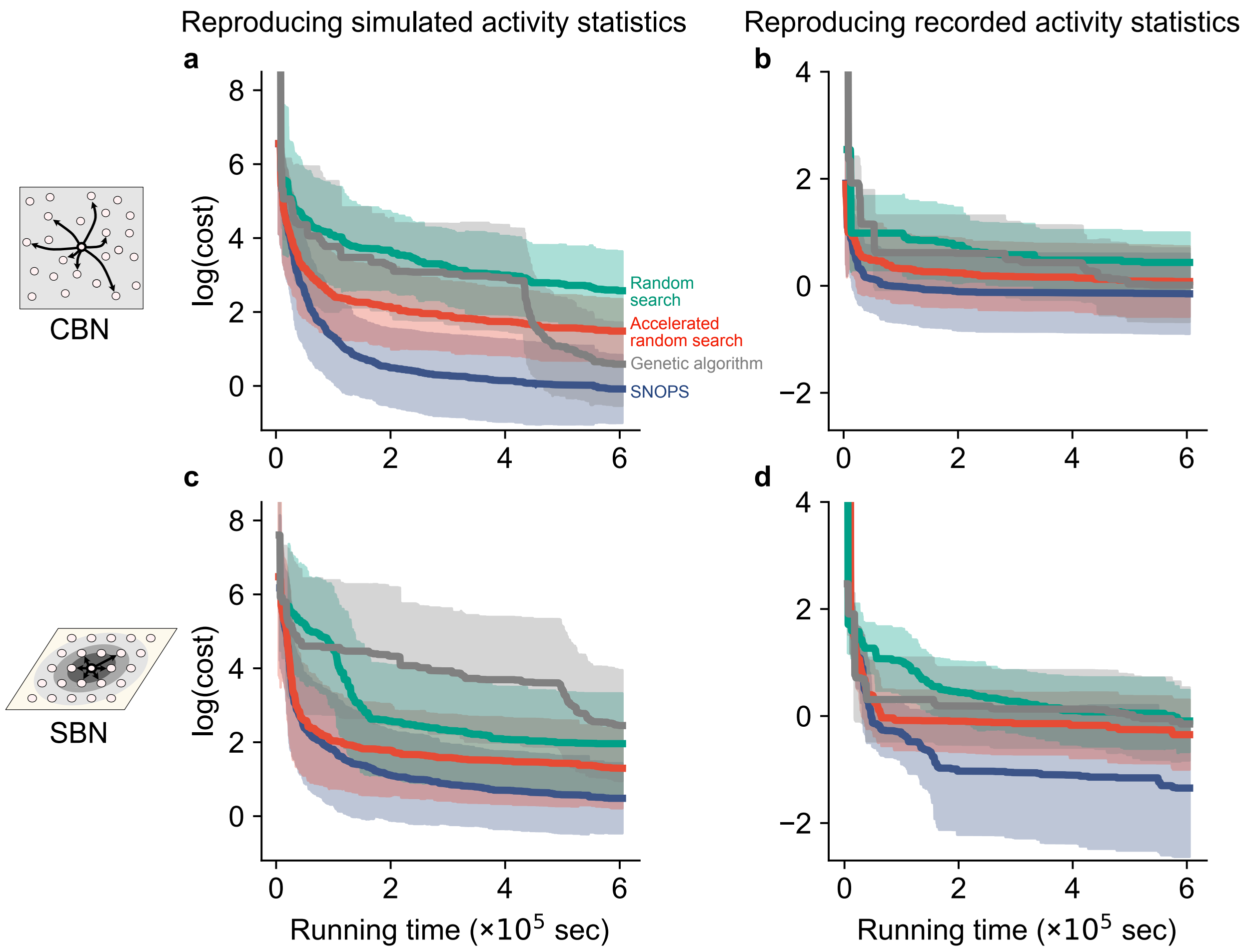
We provided an overview of Bayesian optimization in SNOPS in Fig. 3. Here we demonstrate the steps in more detail, in particular those shown in Figs. 3d and 3e. For illustrative purposes, we consider the optimization of a single model parameter θ.

Details: We first generated spike trains from an SBN whose connection strength, J^{eF} (see Methods), was set to 60 mV. The values of the rest of the model parameters are set as: $\tau^{id} = 8$ ms, $\tau^{ed} = 5$ ms, $J^{ei} = -60$ mV, $J^{ie} = 10$ mV, $J^{ii} = -75$ mV, $J^{ee} = 20$ mV, $J^{iF} = 25$ mV, $\sigma^i = 0.1$ mm, $\sigma^e = 0.1$ mm. The goal of SNOPS here is to customize a separate SBN to these generated spike trains, where J^{eF} is the only model parameter that is unknown. Because this illustration applies to any model parameter, we denote J^{eF} as θ and the ground truth value of J^{eF} as $\theta^\star$.

a, A Gaussian process (GP) is used to approximate the cost function, $\hat{c}(\theta)$. Same conventions as in Fig. 3d. With each iteration of SNOPS (left to right), we evaluate the cost at one additional value of θ (dashed red line). This leads to an update of both the posterior mean (black line) and posterior uncertainty (gray shading) of the GP, which represents an interpolation of the evaluated costs. Note that the black line might not necessarily pass through all the dots due to the smoothness assumption of the GP, where the smoothing is determined by the Matérn 5/2 kernel (see Methods).

b, A separate GP represents the feasibility function, $\hat{g}(\theta)$. For example, a value of θ would be infeasible if it leads to an unrealistically high firing rate (see *Feasibility constraints*). The rationale for using a GP is that, if a value of θ is feasible (or infeasible), similar values of θ are also likely to be feasible (or infeasible). A parameter θ is feasible if $\hat{g}(\theta) = 1$ and infeasible if $\hat{g}(\theta) = 0$. As in **a**, the GP is updated with each iteration of SNOPS (left to right) based on whether the evaluated θ is feasible or infeasible (black dots). Note that, for $t = 4$, there are three dots in **a**, but four dots here in **b**. The reason is that the rightmost dot (representing the iteration before $t = 4$) corresponds to an infeasible value of θ (i.e., $\hat{g}(\theta) = 0$), and hence the cost at that θ was not evaluated.

c, Based on the predicted cost function (from **a**) and feasibility function (from **b**), SNOPS computes an acquisition function, $a(\theta)$. The maximum of the acquisition function (dashed red line) determines the θ to evaluate during the next iteration of SNOPS. Intuitively, the acquisition function identifies values of θ that are feasible and for which the predicted cost is low. In addition, the acquisition function implements an exploration-exploitation trade-off, whereby values of θ for which the cost is uncertain are favored. As SNOPS iterates (left to right), the maximum of $a(\theta)$ approaches the ground truth value $\theta^\star$.



Supplementary Figure 3: Comparing SNOPS, random search methods, and a genetic algorithm on simulated activity and neuronal recordings

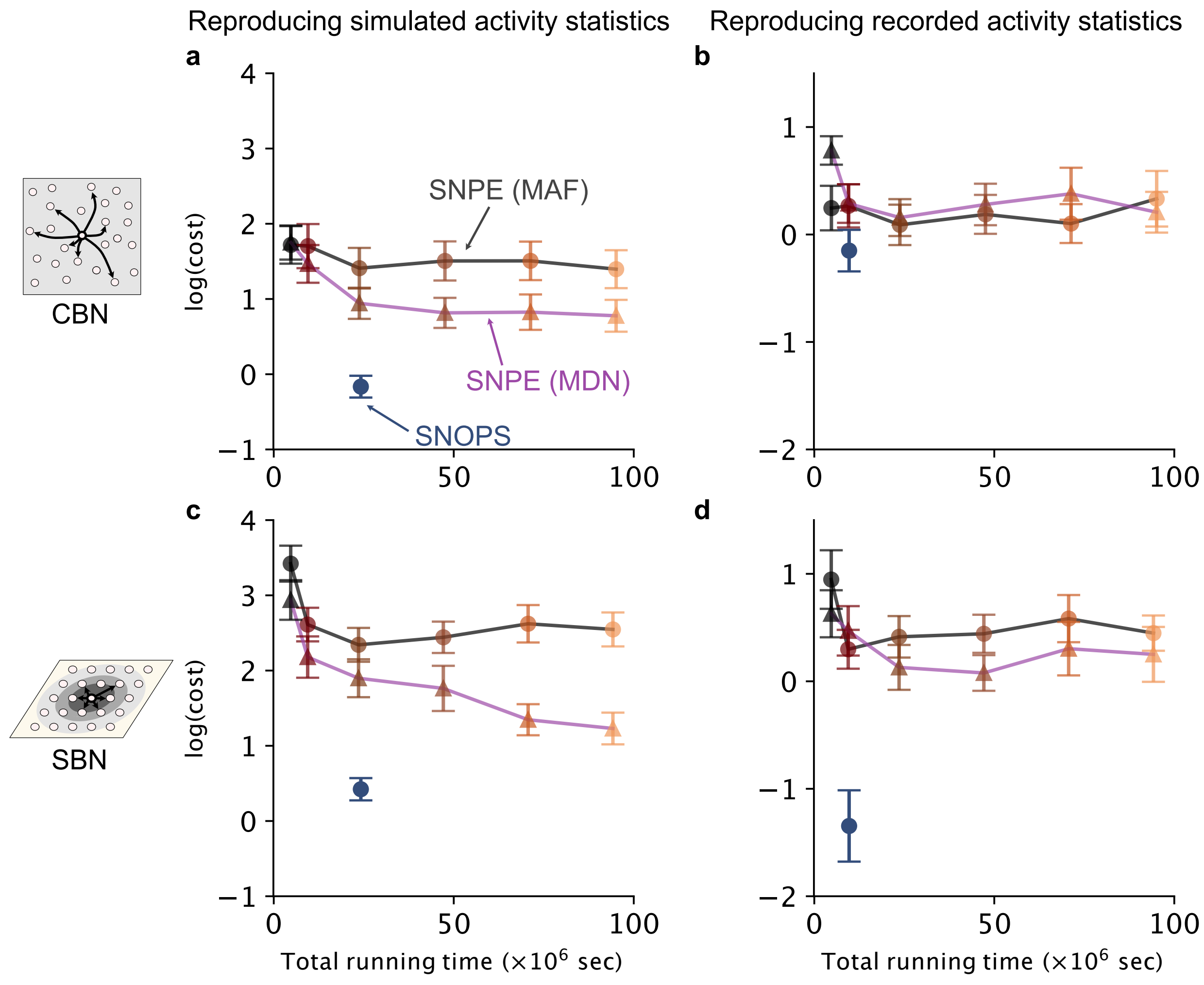
In Fig. 4, we showed that SNOPS outperforms random search methods when customizing a CBN to simulated activity. Here we compare SNOPS, random search, accelerated random search, and a genetic algorithm[2–4] in a wider range of settings, involving also SBN and neuronal recordings. Same conventions as Fig. 4b. **a,** Customizing a CBN to simulated activity. These curves are identical to those shown in Fig. 4b. **b,** Customizing a CBN to neuronal recordings in macaque V4 and PFC. These curves correspond to the results for the CBN in Fig. 5c. **c,** Customizing a SBN to simulated activity. **d,** Customizing a SBN to neuronal recordings in macaque V4 and PFC. These curves correspond to the results for SBN in Fig. 5c.

For the genetic algorithm (GA), we used the Matlab function ga with the same search range and total running duration as the other algorithms. Using the function's default configuration, we set the population size (the number of parameter sets evaluated in each population) to 200, the elite count (the number of parameter sets that are guaranteed to survive into the next generation) to 10, and the cross-over fraction (the fraction other than the elite children that are produced by crossover) to 0.8.

Each curve indicates how the cost (which we seek to minimize, plotted in log scale) changes during the optimization procedure, as a function of the computer running time. The shading represents mean ± 1 SD across the customization runs. Panels **a** and **c** include 40 customization runs, as in Fig. 4. Panels **b** and **d** include 16 customization runs corresponding to the neuronal recordings (2 monkeys $\times$ 2 brain areas $\times$ 4 experimental conditions) in Fig. 5c. Note that 10^5 seconds equals 1.2 days or 27.8 hours.

The cost decreases with running time for all four algorithms on both simulated activity and neuronal recordings, as expected. The cost plateaus for all algorithms, suggesting that the stopping criterion we set (see Methods) is sufficient for the algorithms to converge. We found that SNOPS outperforms the random search methods and GA for both simulated data and recorded activity for both CBN and SBN (the blue curves are below the red, green, and grey curves). This result demonstrates that Bayesian optimization enables SNOPS to find a better solution than random search methods and GA for a given amount of computer running time. Accelerated random search also outperforms random search across datasets and models (the red curves are below the green curves), suggesting that the two innovations (feasibility constraint and intensification, see Methods) further accelerate the customization process.

The amount by which Bayesian optimization outperforms the random search methods depends on the specific network model and the activity to which the model is being customized. For example, SNOPS performed similarly to accelerated random search in **b**. This likely occurs because the CBN is unable to generate a wide range of population statistics, and so there are many parameter sets that correspond to a similar minimal cost. As a result, the random search methods can readily find one of these parameter sets. Even so, SNOPS performed as well or better than the other two methods.



Supplementary Figure 4: Comparing SNOPS and SNPE on simulated activity and neuronal recordings

In Fig. 4 and Supplementary Fig. 3, we showed that SNOPS outperformed random search methods when customizing both CBN and SBN to simulated and recorded neuronal activity. Here we compare SNOPS to a method that uses deep-learning-based inference, Sequential Neural Posterior Estimation (SNPE)[5], using the same models and neuronal recordings as in Supplementary Fig. 3. One might also consider using Emergent Property Inference (EPI)[6]. However, EPI is not applicable to large-scale SNNs because it requires the cost function to be differentiable with respect to the model parameters.

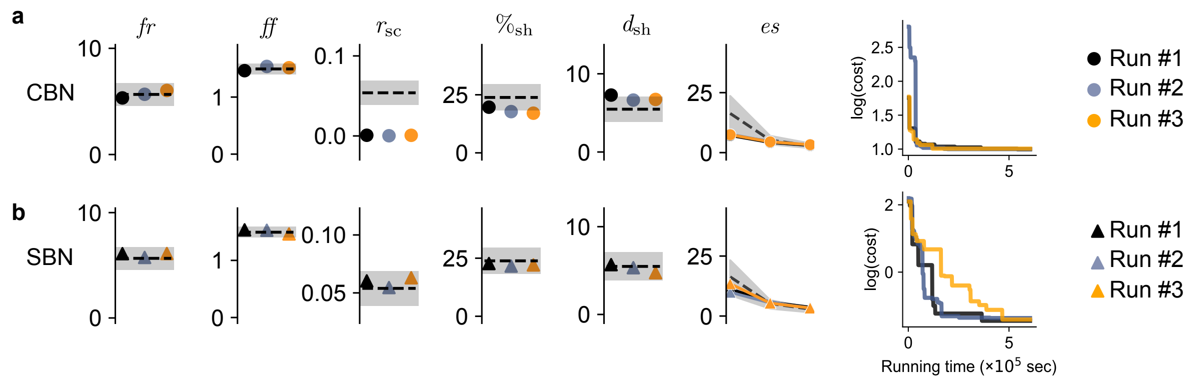
SNPE uses deep neural networks to approximate the distribution of the model parameters given the recorded activity statistics. Applying SNPE involves a training stage and an inference stage. During the training stage, a deep generative model is trained on a large number of simulations comprising the ground truth parameter set of each simulation and its corresponding activity statistics. During the inference stage, given the recorded activity statistics, SNPE returns samples of parameter sets from a posterior distribution representing the likelihood of the parameters consistent with the recorded activity. We wish to compare the two methods in terms of their optimal cost and total running time. There are two key differences between SNOPS and SNPE: (1) SNOPS returns a single parameter set whereas SNPE returns a distribution of parameter sets. The choice between these two outputs depends on the number of customized models desired for the scientific goal (see Discussion). (2) SNPE requires a large number of computationally demanding simulations for its training stage. It can then perform fast parameter inference on any new dataset without the need to run additional simulations. In contrast, SNOPS does not need simulations upfront. However, for each new dataset, SNOPS requires a separate customization run consisting of computationally demanding simulations in the iterative procedure.

Comparing the two methods fairly is challenging because they are tailored for different use cases (see Discussion). We did the following to ensure fairness to the best of our abilities (see Details below): (1) To compare the parameter distribution inferred by SNPE to a single parameter set returned by SNOPS, we used the mode of the SNPE distribution as the optimal parameter set for SNPE. (2) For the total running time of SNOPS, we considered the time required for a single customization run, which is equivalent to one dataset, and multiplied it by the number of datasets. Each customization run takes 168 hours to converge, as illustrated in Fig. 3. Therefore, the total running time for SNOPS would be 40×168 hours for the simulated activity and 16×168 hours for the neuronal recordings. For SNPE, the total running time includes two parts. The first part consists of simulating and computing activity statistics using a large number of parameter sets (training size) and training deep networks to capture this relationship. We systematically varied the training size from $5,000$ to $100,000$ simulations to examine its impact on the fitting quality of SNPE. The second part is the inference time for the 40 simulated datasets (panels **a** and **c**) or the 16 recorded datasets (panels **b** and **d**). For both methods, all other settings, including the size of the network model and the datasets to fit, were identical.

The black-to-orange dots represent the cost of SNPE (in log scale) under different training sizes. We considered two neural density estimators for SNPE: masked autoregressive flow (MAF) and mixture density networks (MDN). The blue dots represent the cost and total running time of SNOPS. The error bars represent mean $\pm$ 1 SEM across the datasets: panels **a** and **c** include 40 simulated datasets generated by CBNs or SBNs, as in Supplementary Fig. 3a and c; panels **b** and **d** include 16 datasets of neuronal recordings (2 monkeys $\times$ 2 brain areas $\times$ 4 task conditions), as in Supplementary Fig. 3b and d. Note that 10^5 seconds equals 1.2 days or 27.8 hours.

We found that SNOPS outperforms SNPE: SNOPS (dark blue dot) has a lower cost than SNPE (black-to-orange dots) for different SNPE training sizes and both neural density estimators. The main reason is that SNPE requires a large number of computationally demanding simulations to ensure that the deep neural networks can accurately characterize the relationship between model parameters and activity statistics. SNPE may need more than $100,000$ training samples to achieve comparable results to SNOPS. However, the high computational cost for SNPE can be justified by the benefits of a richer characterization of the parameter space since it returns a distribution of parameter sets and provides fast inference for new datasets. Therefore, the choice between SNOPS and SNPE depends on the network simulation time and the goal of model customization. When the simulation time is low, SNPE can be used to obtain a parameter distribution and make fast inferences on a large number of new datasets. In this case, SNOPS might incur a greater running time because SNOPS starts from scratch for each new dataset. When the simulation time of the model is substantial (as for the CBN and SBN), generating a sufficient amount of training samples for SNPE can be computationally prohibitive. In this case, one can instead use SNOPS to obtain a single optimal parameter set for a given dataset. It is also noteworthy that, with computational resources that support parallelization, such as GPUs or a large array of CPUs, the actual running time (clock time) for SNPE could be substantially reduced. This efficiency gain is due to the ability to run a large number of simulations in parallel during the generation of training samples for SNPE. Conversely, SNOPS is not as easy to parallelize due to its iterative nature, which relies on the outcomes of previous iterations to determine the parameter set for subsequent iterations (although see Methods for a way to partially parallelize SNOPS).

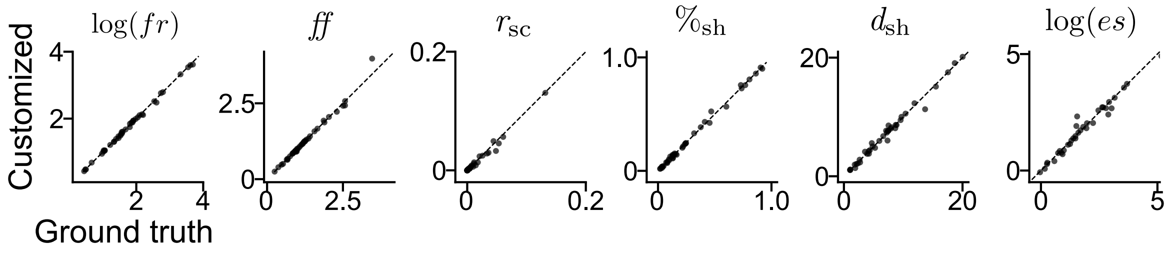
Details: For SNOPS, the cost for each dataset (simulated or recorded) is the result of one customization run (Supplementary Fig. 3, cost values at the right end of the blue curves). For SNPE, we trained a deep neural model using different training sizes: 5,000, 10,000, 25,000, 50,000, 75,000, and 100,000. For each training size, we first sampled the corresponding number of parameter sets according to a uniform distribution, which is the default setting in SNPE. For each sampled parameter set, we generated spike trains under one connectivity graph and computed its activity statistics. We then fed the sampled parameter sets, as well as their activity statistics, to the SNPE training algorithm and ran it until convergence. We did not include training sizes larger than 100,000 because it would require over $25,000$ CPU hours, which was computationally prohibitive. At the inference stage, for a set of activity statistics corresponding to a simulated or recorded dataset, we drew $10,000$ parameter sets from the SNPE posterior distribution. We selected the single parameter set with the largest log-likelihood returned by SNPE, representing the mode of the posterior distribution. We computed the cost for SNPE using five network instantiations of connectivity graphs and initial membrane potentials corresponding to the same identified parameter set in the same way as in SNOPS.



Supplementary Figure 5: Consistent results from SNOPS across multiple runs

In Fig. 5, we compared the performance of the CBN and SBN in reproducing the activity statistics of recordings in area V4 and PFC. Here we test the reliability of the results returned by SNOPS. Ideally, we would assess this reliability by ensuring that it returns the "ground truth" parameters for each recording. However, because the real neuronal recordings were not generated by a model, there are no "ground truth" parameters. Instead, we assessed whether varying the initialization of SNOPS would affect results. There were several initialization settings that could affect the outcome of the customization process: the initially sampled seed parameter sets for SNOPS ($\hat{\Theta}$ in Algorithm 2), the randomly generated connectivity graph, the randomly configured initial membrane potentials for each neuron, and the stochasticity of the activity in the feedforward layer (see Methods).

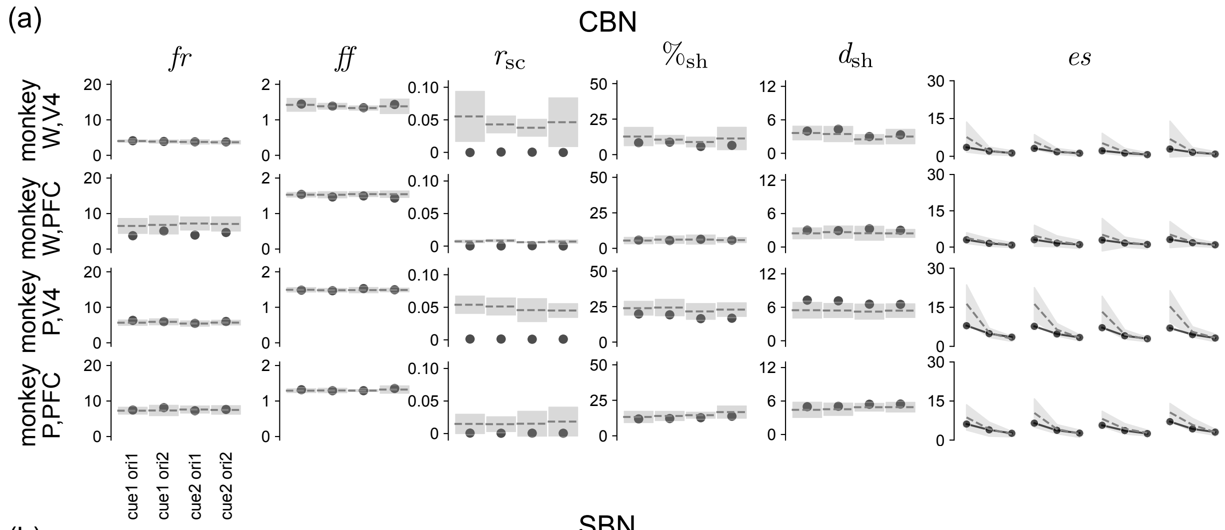
To investigate if SNOPS returned consistent results, we applied SNOPS to customize CBNs (panel **a**) and SBNs (panel **b**) to the same V4 dataset (monkey P, cue into receptive fields, 45-degree stimulus orientation) as in Fig. 5a and 5b for three customization runs (three colors). The V4 activity statistics are shown as the mean $\pm$ 1 SD across 19 recording sessions. For each customization run, the settings of SNOPS (initially sampled parameters, connectivity graph, initial membrane potentials, activity in the feedforward layer) were randomized (see Methods for the ranges of the values). We found that, regardless of the initialization, SNOPS returned highly reliable results across the three runs (three dots have similar values for all panels). The costs of the customized parameters were also similar across the three runs: $2.72, 2.72, 2.74$ for the CBN; $0.24, 0.26, 0.24$ for the SBN. Hence, SNOPS is robust to different initializations of the optimization procedure. We also include the cost-time traces for each customization run. Note that 10^5 seconds equals 1.2 days or 27.8 hours.

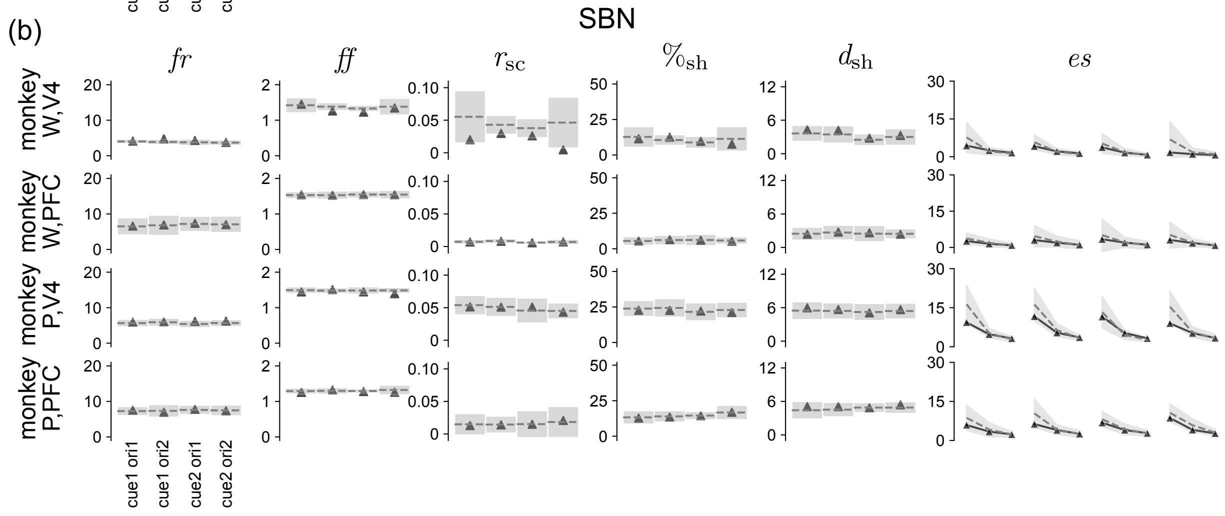


Supplementary Figure 6: Accurate recovery of ground truth activity statistics with SBNs

In Fig. 4, we validated the performance of SNOPS in customizing the CBN to simulated activity. Here we perform the same analysis with the SBN. As we did with the CBN, we randomly drew 40 parameter sets for the SBN and used them to simulate spiking activity. We then used SNOPS to customize a SBN to the simulated activity (see *Verifying SNOPS in simulation*). Same conventions as in Fig. 4d.

We found that SNOPS was able to find model parameters for the SBN that accurately reproduced the activity statistics (all dots lie near the diagonal line). For r_{sc}, one data point (0.65, 0.68) fell outside of the plotting range and is not shown.



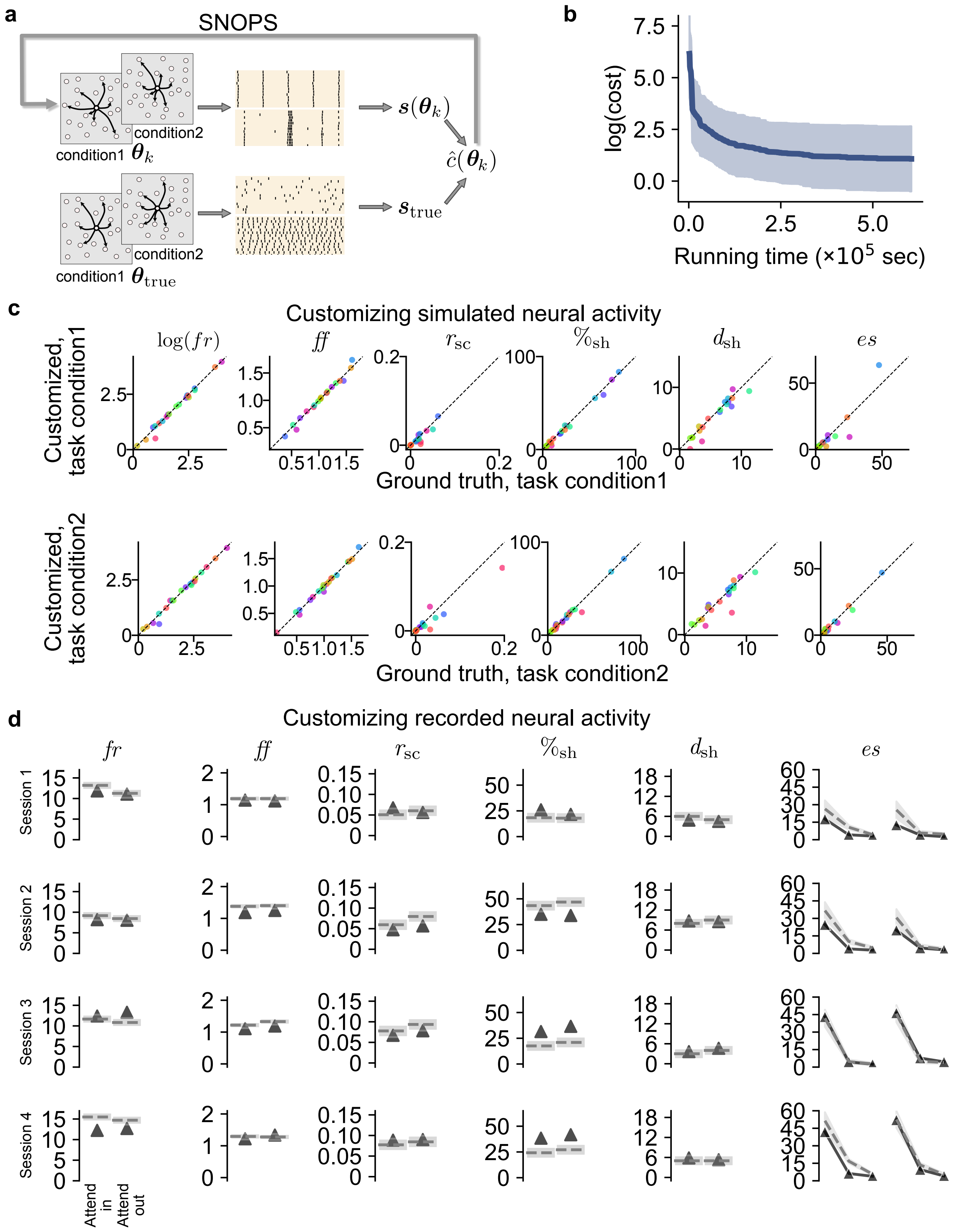


Supplementary Figure 7: Activity statistics of the network models customized by SNOPS to neuronal recordings

In Fig. 5c, we compared the performance (in terms of the cost) of the SBN and CBN on 16 datasets spanning two monkeys, two brain areas, and four task conditions. Here we show the activity statistics of the SNOPS-customized CBN models (panel **a**, circles, mean across 5 network instantiations corresponding to the same identified parameter set) and SBN models (panel **b**, triangles, mean across 5 network instantiations corresponding to the same identified parameter set) for the same 16 datasets as in Fig. 5c. Dashed lines and shadings represent the mean $\pm$ 1 SD across recording sessions (see Methods for the number of recording sessions for each monkey and brain area).

SNOPS was able to find parameter sets for SBN that are close to the recorded statistics (triangles are within or close to the grey shading). For the CBN, the r_{sc} did not match those of the neuronal recordings, consistent with Fig. 5a. The example datasets shown in

Fig. 5a and b correspond to monkey P, V4, cue1 ori1 (attention directed to the aggregate V4 receptive field with a 45° stimulus orientation).


a
SNOPS
condition1 $\boldsymbol{\theta}_k$
condition2
$\boldsymbol{s}(\boldsymbol{\theta}_k)$
$\hat{c}(\boldsymbol{\theta}_k)$
condition1 $\boldsymbol{\theta}_{\text{true}}$
condition2
$\boldsymbol{s}_{\text{true}}$
b
log(cost)
Running time (×10^5 sec)
c
Customizing simulated neural activity
$\log(fr)$
ff
r_{sc}
$\%_{\text{sh}}$
d_{sh}
es
Customized, task condition1
Ground truth, task condition1
Customized, task condition2
Ground truth, task condition2
d
Customizing recorded neural activity
fr
Session 1
Session 2
Session 3
Session 4
Attend in
Attend out

Supplementary Figure 8: Customizing network models for multiple task conditions using SNOPS

In Figs 4-6, we used SNOPS to customize a network model to spontaneous activity (i.e., a single task condition). Here, we consider using SNOPS to customize a network model that describes two different task conditions. In this model, some model parameters have the same values across task conditions, whereas other parameters can have different values across task conditions.

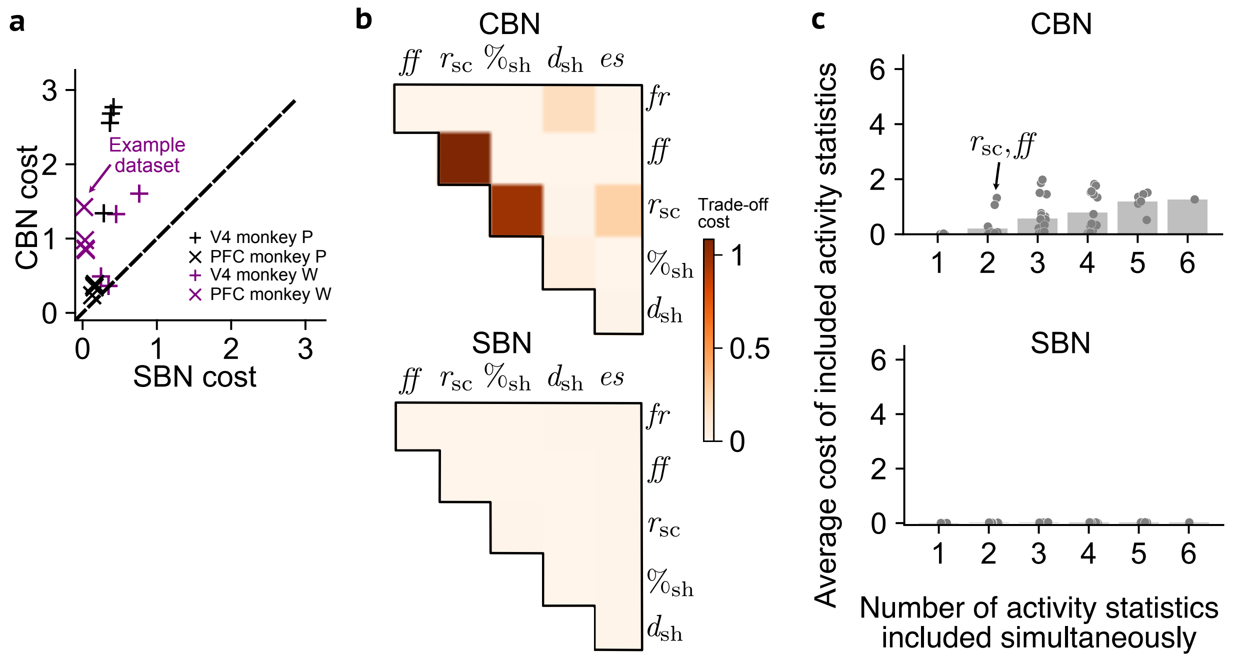
We constructed a model comprising two SBNs, which share the same values for 11 parameters (those shown in Supplementary Tables 1 and 2) and have different values for two new parameters (μ_E and μ_I, which represent external static currents to excitatory and inhibitory neurons, respectively). This unified model has a total of 15 parameters. This setup allows the unified model to represent two task conditions, such as different attentional states. The shared parameters pertain to network structure, whereas the unique parameters relate to external inputs that can be different for each task condition. The search ranges for the new parameters were set to $[-1, 1]$; those for the other parameters remained consistent with Supplementary Tables 1 and 2.

We first verified that SNOPS can successfully customize the unified model to simulated activity. Similar to Fig. 4, we randomly sampled 1000 ground truth parameter sets from the 15-dimensional parameter space and simulated each network. For computational expediency, we used a smaller network model ($N_F = 2,500, N_e = 1,600, N_i = 400$) for this analysis. To make the simulation as realistic as possible, we focused on 20 simulated datasets whose fr was larger than 1, ff (averaged across neurons) was within the range $[0, 2]$, and r_{sc} was within the range $[0, 0.2]$. We then used SNOPS to customize a unified network model (with 15 free parameters) to the activity statistics of the simulated activity (panel **a**). Each ground truth dataset comprised two sets of activity statistics corresponding to the two task conditions. During customization, network simulations were run for each component SBN, with the overall cost being the average cost across the two-component SBNs.

SNOPS converges within 168 hours (panel **b**, the curve plateaus, note that 10^5 seconds equals 1.2 days or 27.8 hours). The SNOPS-customized models successfully generated activity statistics that closely matched the ground truth (panel **c**, dots located near diagonal). Each row represents one task condition (i.e., one component SBN) of the unified model. Dots of the same color correspond to the same customization run; the fr was plotted in log scale for visual clarity. Notably, the activity statistics generated by the network were different across the two task conditions (panel **c**, dots of the same colors are positioned differently across the two rows), mirroring the scale of differences observed in real data (see below). This was achieved with only two differing parameters between the component SBNs.

We then customized the unified model to four representative sessions from monkey P, focusing on different attentional states in V4 (panel **d**). The dashed lines represent the activity statistics for each session, and the shading represents 1 SD across 19 recording sessions of the same monkey and brain area. We combined the trials from both orientations for each session.

In the task, the monkey deployed its visual attention either into ('attend in') or outside ('attend out') of the receptive fields of the recorded neurons (see Methods). Note that in the main text, we analyzed these two attention conditions separately; here we use the unified model to capture the activity statistics of the two attention conditions together. The activity statistics for both attention conditions were computed using spike counts taken using a 200 ms bin, starting 50 ms after the onset of the visual stimulus (when the neurons start to respond to the stimulus). Consistent with previous work[7,8], the 'attend in' condition exhibited a higher fr and lower r_{sc} compared to the 'attend out' condition. SNOPS successfully customized the unified model to the neuronal recordings (triangles near dashed lines), with slightly larger errors compared to customizing a standard SBN to activity from one task condition (cf. Fig. 5 and Supplementary Fig. 7). The unified model needed to capture the differences in activity statistics across the two task conditions by varying only two model parameters (and sharing 11 task parameters), representing a more difficult customization task than when customizing a standard SBN to activity from a single task condition. Despite these challenges, SNOPS was able to customize a unified network model to capture activity statistics across multiple task conditions.



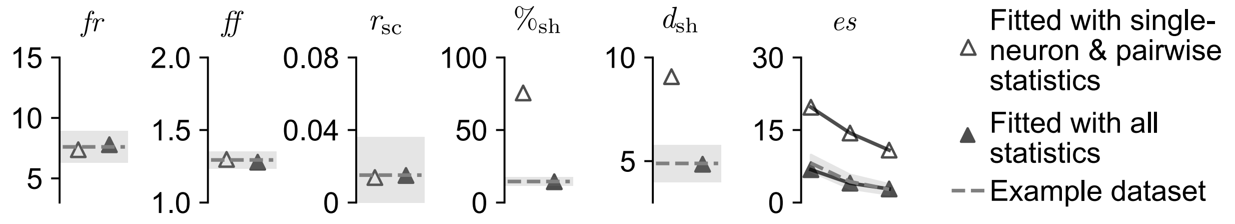
Supplementary Figure 9: Trade-off costs for CBN and SBN when customizing to PFC activity

In Fig. 6, we examined the trade-offs between activity statistics for the CBN and SBN on an example V4 dataset. Here we repeated the analysis with an example PFC dataset. Same conventions as in Fig. 6.

a, Reproduction of Fig. 5c, but the arrow now indicates a PFC dataset that is examined more closely here in **b** and **c**.

b, Trade-off costs between pairs of activity statistics for the CBN and SBN on the example PFC dataset indicated in **a**. We found that, for the CBN, r_{sc} has a high trade-off cost with ff and $\%_{sh}$, as was seen for V4 (cf. Fig. 6c). However, the size of this trade-off was less pronounced in PFC relative to V4. To understand this, PFC exhibits lower r_{sc} values (0.006 ± 0.002) than V4 (0.054 ± 0.015). This makes it easier for the CBN to reproduce the r_{sc} while concurrently reproducing the other activity statistics.

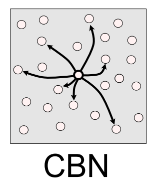
c, Customizing the CBN and SBN to different numbers of activity statistics included in the cost function simultaneously. This was performed using the example PFC dataset indicated in **a**. As in V4 (cf. Fig. 6d), we found that the average cost of the customized activity statistics increases as more statistics are included for the CBN. By contrast, the average cost for the SBN remains low for even up to all six simultaneously customized statistics.

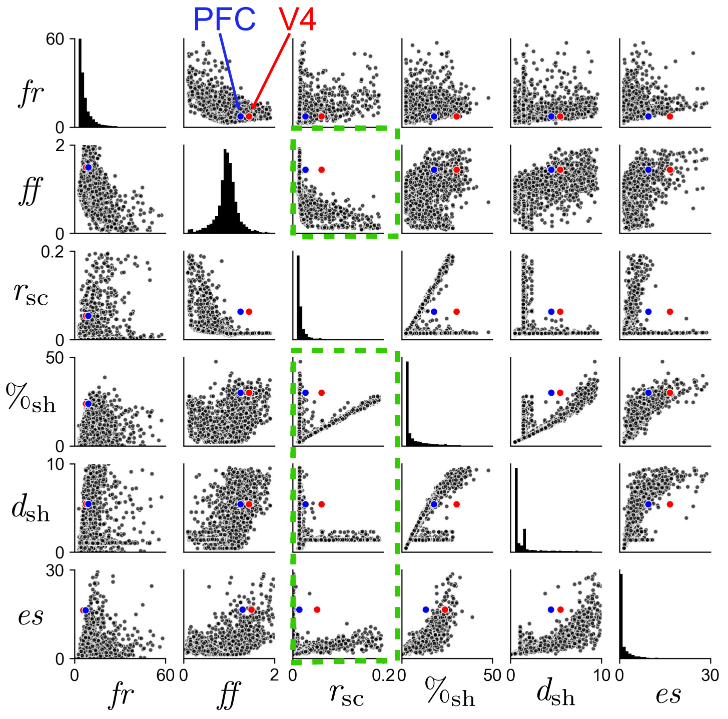


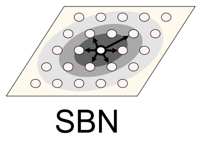
Supplementary Figure 10: Benefits of including population statistics in the cost function

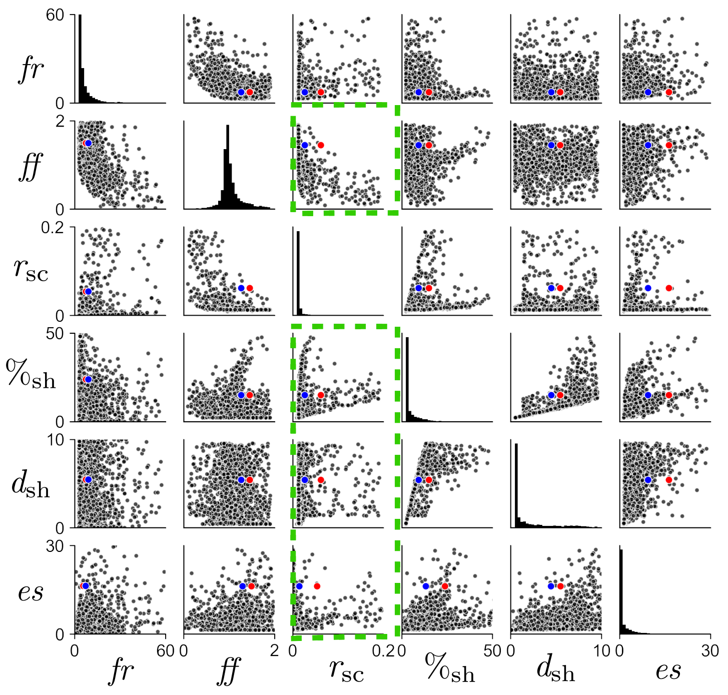
In Fig. 6, we found that the SBN is more flexible than the CBN in reproducing activity statistics. When a model is more flexible, it might need to be more strongly constrained during the customization process to accurately reproduce the activity statistics. Here, we ask if the SBN model is well-constrained enough such that simply customizing single-neuron and pairwise statistics will automatically lead to accurately reproducing the population statistics.

We applied SNOPS to customize the SBN to an example PFC dataset (monkey P, cue into receptive fields, 45-degree stimulus orientation) under two scenarios: (1) only single-neuron and pairwise statistics were included in the cost function (open triangles), and (2) all single-neuron, pairwise, and population statistics were included in the cost function (filled triangles). The PFC activity statistics are shown as the mean $\pm$ 1 SD across 19 recording sessions. Although the single-neuron and pairwise statistics were accurately reproduced in both scenarios, leaving the population statistics out of the cost function led to a poor match in population statistics (open triangles are far from the dashed lines). Hence the inclusion of population statistics in the cost function is necessary for constraining the SBN during the customization process. More generally, customizing models even more flexible than the SBN might necessitate the inclusion of additional activity statistics in the cost function.









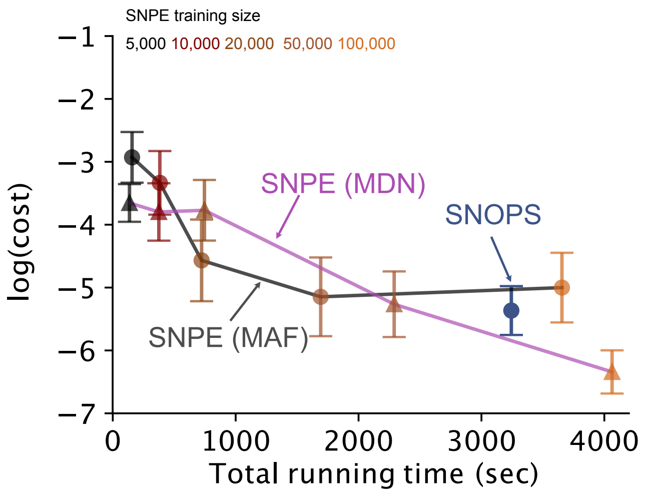
(Continued on the following page.)

Supplementary Figure 11: Comparing the achievable combinations of activity statistics for SBN and CBN

In Fig. 6, we assessed each model's flexibility by asking whether it could simultaneously reproduce two or more activity statistics. Here we visualize the range of combinations of activity statistics that each model is able to generate.

We randomly drew $5,000$ parameter sets from the search range of each model (see *Spiking network models*). For each parameter set, we simulated activity with three network instantiations, and computed the average activity statistics across the instantiations. Each dot represents a pair of activity statistics for one randomly sampled parameter set. For illustrative purposes, we only show the parameter sets whose activity statistics fall within the following ranges: $fr : [0, 60], ff : [0, 2], r_{\mathrm{sc}} : [0, 0.2], \%_{\mathrm{sh}} : [0, 50], d_{\mathrm{sh}} : [0, 10], es : [0, 30]$. The red and blue dots represent the example V4 and PFC activity statistics in Fig. 5. The vertical axes of the histograms (subpanels along the diagonal) represent the proportion of dots rather than the vertical axes indicated.

The dots visually occupy a greater area for the SBN than CBN, indicating that the SBN is capable of generating more diverse pairs of statistics than the CBN. This is particularly prominent when comparing the CBN and SBN for the subpanels highlighted by the dashed green boxes. In these subpanels, the V4 statistic pairs (red dots) lie outside the range of the statistic pairs generated by the CBN (black dots, upper plot) but inside the range of those generated by the SBN (black dots, lower plot). This demonstrates that the CBN is incapable of simultaneously reproducing r_{sc} and any one of the following statistics – ff, $\%_{\mathrm{sh}}$, d_{sh}, or es – of the V4 dataset. This is consistent with the activity trade-off costs identified in Fig. 6c. The visualizations here provide further evidence that the SBN is more flexible than CBN in producing multiple activity statistics simultaneously.

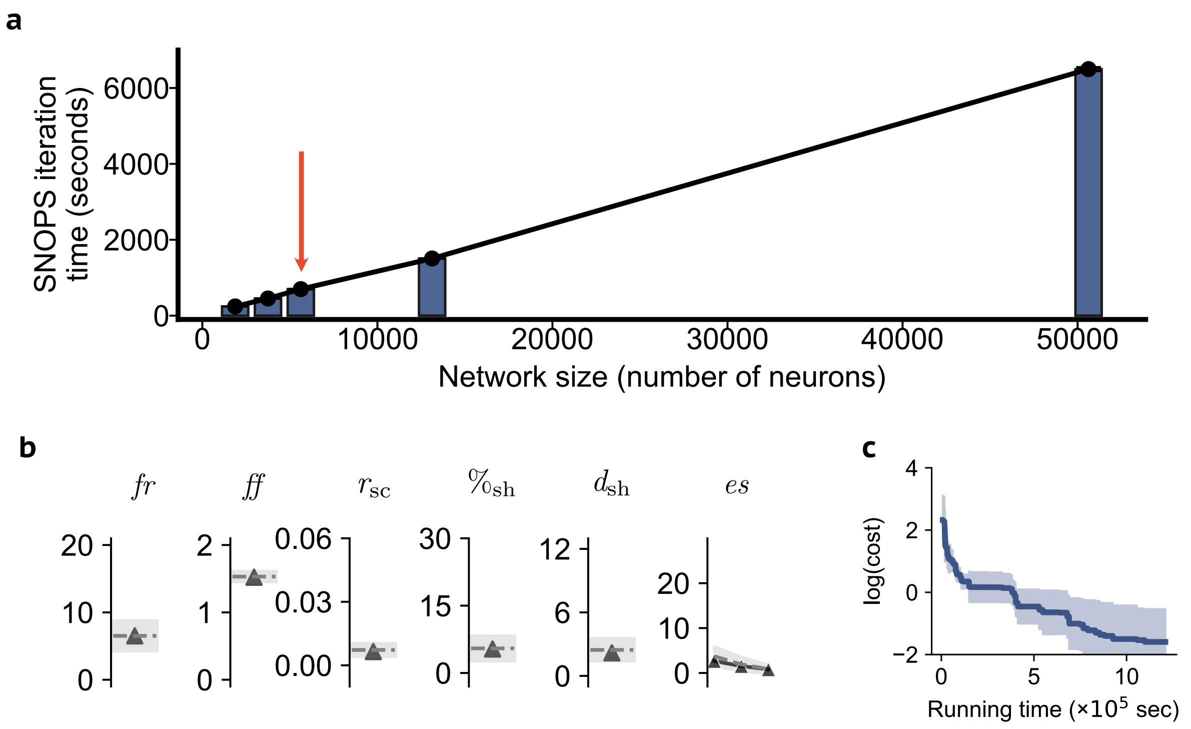


Supplementary Figure 12: Similar performance of SNOPS and SNPE for the Hodgkin-Huxley model

In Supplementary Fig. 4, we compared SNOPS with SNPE in customizing the large-scale spiking network models (SBN and CBN). Here we compare the performance of SNOPS with SNPE in customizing a biophysical single-neuron model: the Hodgkin-Huxley (HH) model[9]. The simulation time of the HH model is three orders of magnitude less than that of the large-scale spiking network models. For the HH model simulator, there are eight free parameters (ranges are indicated in brackets): the sodium conductance per unit area ($g_{Na} \in [0.5, 80]$), the sodium reversal potential ($E_{Na} \in [1, 300]$), the potassium conductance per unit area ($g_K \in [10^{-4}, 15]$), the potassium reversal potential ($E_K \in [-200, -1]$), the leak conductance per unit area ($g_l \in [10^{-4}, 0.6]$), the leak reversal potential ($E_l \in [35, 100]$), the resting potential ($V_0 \in [-100, -25]$), and the amplitude of the injected current ($I_{inj} \in [0, 0.5]$). The HH model was simulated for 120 seconds with a step size of 0.01 second, and the current was injected at the 20th second. Similar to the analyses in Goncalves et al., 2020, we analyzed seven statistics related to spike generation: the number of spikes, the mean of the resting potential, the standard deviation of the resting potential, and the mean, standard deviation, skewness, and kurtosis of the voltage after current injection. To obtain ground truth datasets, we randomly sampled 20 sets of model parameters, simulated the HH model using those parameters, and computed the statistics resulting from the simulations.

For SNOPS, we customized a HH model to each of the 20 sets of spike generation statistics. For each set of statistics, SNOPS was run for 150 iterations. We did not use the intensification procedure because each run of the HH model yields exactly the same statistics. For SNPE, we generated large numbers (5,000, 10,000, 20,000, 50,000, 10,000) of model parameters-to-statistics relationships for training the neural density estimators. As in Supplementary Fig. 4, we used the mode of the posterior distributions as the optimal parameter set for SNPE. We compared SNOPS to two variants of SNPE, each with a different neural density estimator: the masked autoregressive flow (MAF) and the mixture density network (MDN). The total running time for SNOPS is the sum of the running times of the 20 customization runs. The total running time for the SNPE includes running a (large) number of HH simulations (indicated above by the color of the symbol), training the deep neural network, and performing inference using the trained density estimator. The running times for SNPE-MAF and SNPE-MDN can differ for the same training dataset size due to the difference in time required to train the neural density estimators.

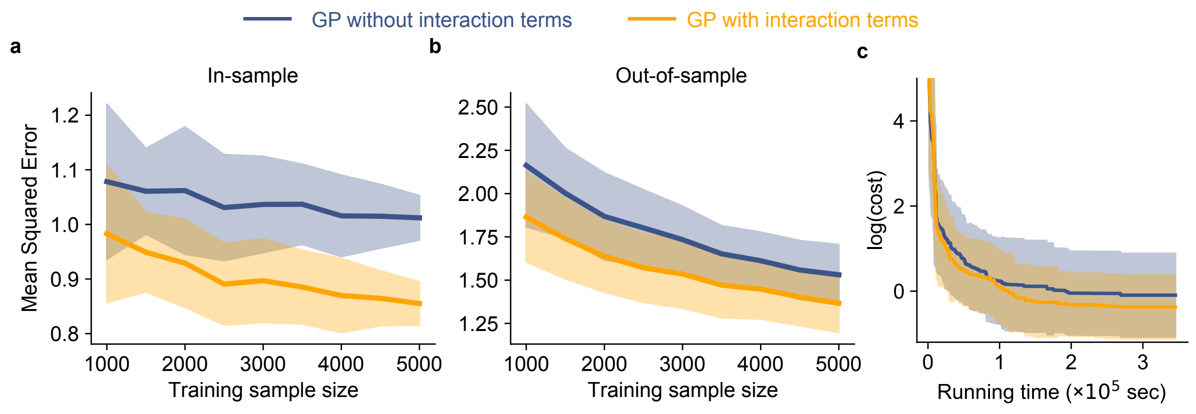
SNOPS performs similarly to SNPE in customizing HH models (the blue dot is near the SNPE traces; error bars represent $\pm$ one standard error). In general, SNPE excels when the simulation time for the model is short (e.g., HH models, simulation time < 0.1 second), which makes it more computationally feasible to generate the large number of simulations for training its deep neural networks. In contrast, SNOPS excels when the simulation time for the model is long (e.g., CBN or SBN, simulation time > 300 seconds, see Supplementary Fig. 4). Even though customizing HH models corresponds to a regime where SNPE excels, SNOPS performs similarly to SNPE.



Supplementary Figure 13: Effect of network size on SNOPS customization

Here we examine the influence of network size on SNOPS' running time and how well SNOPS can reproduce activity statistics. We varied the network size over a wide range: 1,875, 3,750, 5,625 (highlighted with a red arrow, used in our main analyses), 13,125, and 50,625 neurons. For each network size, we customized a SBN to the activity statistics of the V4 dataset in Fig. 5. For each network size, we performed 12 customization runs, each lasting 12 hours to accumulate a sufficient number of SNOPS iterations for our analysis. We found that SNOPS' running time scales almost linearly with network size (panel **a**, mean ± SEM, error bars invisible due to small values). SNOPS' running time is primarily determined by the network simulation time, which varies greatly with network size. The time spent computing activity statistics and fitting Gaussian processes depends little or not at all on the network size.

To investigate if the size of the network model influences its ability to reproduce target activity statistics, we customized a SBN comprising 8,625 neurons (2,500 feedforward neurons, 4,900 recurrent excitatory neurons, and 1,225 recurrent inhibitory neurons) to four example PFC datasets from monkey W. This SBN is larger than the SBN used throughout Results, which comprises 5,625 neurons. We found that the larger SBN successfully reproduced the PFC statistics (see panel **b** for an example dataset) and showed a consistent decrease in the cost over time (panel **c**, cost-time trace across four datasets, shading represents one standard deviation, note that 10^5 seconds equals 1.2 days or 27.8 hours). These results are similar to when we customize a SBN with 5,625 neurons (compare panel **b** to Supplementary Fig. 7, monkey W, PFC). These findings illustrate that our customization results using SNOPS are robust across different network sizes.



Supplementary Figure 14: Benefits of accounting for parameter interactions during SNOPS customization

During SNOPS customization, we fit a Gaussian Process (GP) to capture the relationship between parameter sets and their associated costs. One can build prior knowledge about relationships amongst the parameters into the GP kernel. For example, the ratio between the inhibitory and excitatory time scales, τ_i/τ_e, might influence the r_{sc}[8], and the determinant of the connection strength matrix, $J^{ei} \times J^{ee} - J^{ei} \times J^{ii}$, might influence the fr[10]. Here we explore the potential benefits of incorporating interactions between model parameters into the GP fitting process.

To fit a GP, we need parameter sets and their associated costs. We first randomly sampled 8,000 SBN parameter sets. To compute their costs, we simulated activity from each of these SBNs and computed their costs relative to the following activity statistics (that we obtained from a SBN): $fr = 6.59$, $ff = 1.56$, $r_{\text{sc}} = 0.0003$, $\%_{\text{sh}} = 13$, and $d_{\text{sh}} = 6$. As in the main text, we averaged the estimated costs over 3 network instantiations for each parameter set. We then fit a GP to capture the relationship between the parameter sets and their associated costs under two scenarios: the original setting without parameter interactions (blue curves) and a new setting that includes parameter interactions (orange curves). Specifically, in the GP with interaction terms, we incorporated two additional model 'parameters' into the GP: τ_i/τ_e, and $J^{ei} \times J^{ie} - J^{ee} \times J^{ii}$. This effectively increases the number of model parameters from 11 to 13 by appending the two additional parameters to $\boldsymbol{\theta}$ in Eq. (9). The two new 'parameters' in $\boldsymbol{\theta}$ are always computed as functions of, and therefore remain consistent with, the original 11 parameters in the GP fitting process (note that we do not need to infer $\boldsymbol{\theta}$).

We performed this analysis using varying training sample sizes (500, 750, 1,000, 1,250, 1,500, 2,000, 2,500, 3,000, 3,500, 4,000) with a separate testing set of 500 samples. One sample corresponds to one SBN parameter set. We computed the mean squared error (MSE) of the cost predicted by the GP versus the actual cost obtained from the SBN simulations, on both the training (in-sample) and test (out-of-sample) sets to evaluate the quality of the GP fitting. The in-sample MSE measures if the GP model has the capability to fit the training samples and was computed on 500 randomly-chosen training samples. The out-of-sample MSE measures if the GP model fitting generalizes to unseen samples and was computed on the 500 testing samples. We found a reduction in the MSE, both in-sample (panel **a**) and out-of-sample (panel **b**), when the interaction terms are included. The shading represents mean $\pm$ 1 SD across the 500 training or testing samples.

Having confirmed that incorporating parameter interactions improves the GP fitting, we proceeded to test if this also reduces the final cost of the SNOPS customization process. We first simulated activity using 20 ground truth SBN parameter sets, following procedures similar to those described in 'Verifying SNOPS in simulation'. For computational expediency, here we used a smaller SBN (N_F=2,500, N_e=1,600, N_i=400) than in the main text. We customized SBNs of the same smaller size to the simulated activity. We found that the inclusion of interaction terms reduced the final cost of the customization process (panel **c**, orange curve below blue curve, $p = 0.022$, two-sided paired t-test; shading represents mean $\pm$ 1 SD across the 20 ground truth parameter sets; note that 10^5 seconds equals 1.2 days or 27.8 hours) for a given amount of (finite) running time. This improvement demonstrates the value of incorporating prior knowledge about model parameter interactions into the SNOPS customization procedure.

References

[1] Ryan C Williamson, Benjamin R Cowley, Ashok Litwin-Kumar, Brent Doiron, Adam Kohn, Matthew A Smith, and Byron M Yu. Scaling properties of dimensionality reduction for neural populations and network models. *PLoS computational biology*, 12(12):e1005141, 2016.

[2] Péter Friedrich, Michael Vella, Attila I Gulyás, Tamás F Freund, and Szabolcs Káli. A flexible, interactive software tool for fitting the parameters of neuronal models. *Frontiers in neuroinformatics*, 8:63, 2014.

[3] Kristofor David Carlson, Jayram M Nageswaran, Nikil Dutt, and Jeffrey L Krichmar. An efficient automated parameter tuning framework for spiking neural networks. *Frontiers in neuroscience*, 8:10, 2014.

[4] Werner Van Geit, Michael Gevaert, Giuseppe Chindemi, Christian Rössert, Jean-Denis Courcol, Eilif B Muller, Felix Schürmann, Idan Segev, and Henry Markram. Bluepyopt: leveraging open source software and cloud infrastructure to optimise model parameters in neuroscience. *Frontiers in neuroinformatics*, 10:17, 2016.

[5] Pedro J Gonçalves, Jan-Matthis Lueckmann, Michael Deistler, Marcel Nonnenmacher, Kaan Öcal, Giacomo Bassetto, Chaitanya Chintaluri, William F Podlaski, Sara A Haddad, Tim P Vogels, et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. *Elife*, 9:e56261, 2020.

[6] Sean R Bittner, Agostina Palmigiano, Alex T Piet, Chunyu A Duan, Carlos D Brody, Kenneth D Miller, and John Cunningham. Interrogating theoretical models of neural computation with emergent property inference. *Elife*, 10:e56265, 2021.

[7] Marlene R Cohen and John HR Maunsell. Attention improves performance primarily by reducing interneuronal correlations. *Nature neuroscience*, 12(12):1594–1600, 2009.

[8] Chengcheng Huang, Douglas A Ruff, Ryan Pyle, Robert Rosenbaum, Marlene R Cohen, and Brent Doiron. Circuit models of low-dimensional shared variability in cortical networks. *Neuron*, 101(2):337–348, 2019.

[9] Allan L Hodgkin and Andrew F Huxley. Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *The Journal of physiology*, 116(4):449, 1952.

[10] Robert Rosenbaum and Brent Doiron. Balanced networks of spiking neurons with spatially dependent recurrent connections. *Physical Review X*, 4(2):021039, 2014.