ARTICLE **Communicated by Tilo Schwalger**

Inference of a Mesoscopic Population Model from Population Spike Trains

Alexandre René
arene010@uottawa.ca
Department of Physics, University of Ottawa, Ottawa K1N 6N5, Canada; Max Planck Research Group Neural Systems Analysis, Center of Advanced European Studies and Research (caesar), Bonn 53175, Germany; and Institute of Neuroscience and Medicine (INM-6) and Institute for Advanced Simulation (IAS-6) and JARA-Institute Brain Structure-Function Relationships (INM-10), Jülich Research Centre, Jülich 52425, Germany

André Longtin
alongtin@uottawa.ca
Department of Physics, University of Ottawa, Ottawa K1N 6N5, Canada, and Brain and Mind Research Institute, University of Ottawa, Ottawa K1H 8M5, Canada

Jakob H. Macke
macke@tum.de
Max Planck Research Group Neural Systems Analysis, Center of Advanced European Studies and Research (caesar), Bonn 53175, Germany, and Computational Neuroengineering, Department of Electrical and Computer Engineering, Technical University of Munich, Munich 80333, Germany

Understanding how rich dynamics emerge in neural populations requires models exhibiting a wide range of behaviors while remaining interpretable in terms of connectivity and single-neuron dynamics. However, it has been challenging to fit such mechanistic spiking networks at the single-neuron scale to empirical population data. To close this gap, we propose to fit such data at a mesoscale, using a mechanistic but low-dimensional and, hence, statistically tractable model. The mesoscopic representation is obtained by approximating a population of neurons as multiple homogeneous pools of neurons and modeling the dynamics of the aggregate population activity within each pool. We derive the likelihood of both single-neuron and connectivity parameters given this activity, which can then be used to optimize parameters by gradient ascent on the log likelihood or perform Bayesian inference using Markov chain Monte Carlo (MCMC) sampling. We illustrate this approach using a model of generalized integrate-and-fire neurons for which mesoscopic dynamics have been previously derived and show that both single-neuron and connectivity parameters can be recovered from

Neural Computation **32, 1448–1498** (2020) © 2020 Massachusetts Institute of Technology
https://doi.org/10.1162/neco_a_01292

simulated data. In particular, our inference method extracts posterior correlations between model parameters, which define parameter subsets able to reproduce the data. We compute the Bayesian posterior for combinations of parameters using MCMC sampling and investigate how the approximations inherent in a mesoscopic population model affect the accuracy of the inferred single-neuron parameters.

1 Introduction

Neuron populations produce a wide array of complex collective dynamics. Explaining how these emerge requires a mathematical model that not only embodies the network interactions but is also parameterized in terms of interpretable neuron properties. Just as crucial, in order to draw data-supported conclusions, we also need to be able to infer those parameters from empirical observations. These requirements tend to involve a trade-off between model expressiveness and tractability. Low-dimensional state-space models (Macke et al., 2011; Pandarinath et al., 2018; Pillow et al., 2008; Zhao & Park, 2016) are simple enough to allow for inference but achieve that simplicity by focusing on phenomenology: any mechanistic link to the individual neurons is ignored. Conversely, microscopic mechanistic models with thousands of simulated neurons do provide that link between parameters and output (Hawrylycz et al., 2016; Potjans & Diesmann, 2014); however, this complexity makes the analysis difficult and is limited to networks with highly simplified architectures (Doiron, Litwin-Kumar, Rosenbaum, Ocker, & Josic, 2016; Martí, Brunel, & Ostojic, 2018). Since methods to fit these models to experimental data are limited to single neurons (Mensi et al., 2012), it is also unclear how to set their parameters such that they capture the dynamics of large, heterogeneous neural populations.

To reduce the problem to a manageable size and scale, one can consider models that provide a mesoscopic dynamical description founded on microscopic single-neuron dynamics (Dumont, Payeur, & Longtin, 2017; Nykamp & Tranchina, 2000; Wallace, Benayoun, van Drongelen, & Cowan, 2011). Specifically, we will focus on the model described in Schwalger, Deger, and Gerstner (2017), where neurons are grouped into putative excitatory (E) and inhibitory (I) populations in a cortical column. The key approximation is to replace each population with another of equal size but composed of identical neurons, resulting in an effective mesoscopic model of homogeneous populations. In contrast with previous work on population rate dynamics (Gerstner, 2000; Nykamp & Tranchina, 2000; Wilson & Cowan, 1972), Schwalger et al. (2017) correct their mean-field approximations for the finite size of populations. They are thus able to provide stochastic equations for the firing rate of each population with explicit dependence on the population sizes, neuron parameters, and connectivities between populations (see Figure 1A, top). We use these equations to fit the model to traces of population activity.

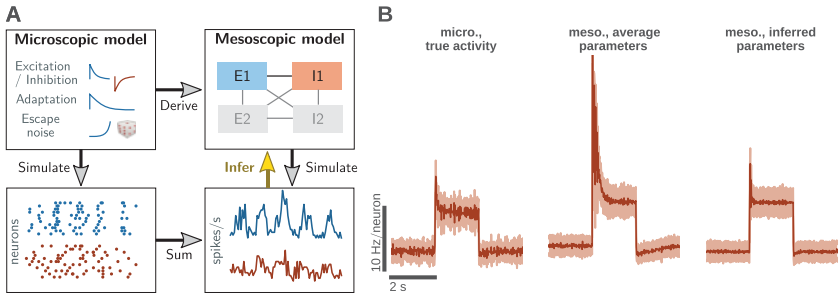

Figure 1: (A) General procedure to infer parameters of a mesoscopic population model from microscopic data. A microscopic model of GIF neurons is used to generate spike trains, which are averaged to obtain traces of population activity; these traces constitute our data. A mesoscopic model of either two or four populations is then fit to these traces. Simulating the mesoscopic model with the inferred parameters allows us to evaluate how well it reproduces the true dynamics. (B) For heterogeneous systems, average parameters might not predict mean activity. Mean activity (line) and its standard deviation (shaded area) for a heterogeneous microscopic model (left) and mesoscopic models attempting to approximate it (middle, right). A mesoscopic model constructed by averaging parameters across the microscopic population overestimates the population's variability (middle). Inferred parameters in this case deviate from these averages and provide a better representation of the true activity (right). Models are as in Figure 5; traces are for the inhibitory population. Means and standard deviations are computed from 50 realizations and averaged over disjoint bins of 10 ms.

Directly inferring mesoscopic model parameters has a number of advantages compared to extrapolating from those obtained by fitting a microscopic model. For one, it allows the use of data that do not have single-neuron resolution. In addition, since neuron parameters in a mesoscopic model represent a whole population, there may not be a clear way to relate micro- and mesoscopic parameters if the former are heterogeneous. By inferring population parameters from population recordings, we target the values that best compensate for the mismatch between the data and the idealized mesoscopic model (see Figure 1B).

The method we present assumes that the model to be inferred can be expressed as a set of stochastic equations and that we have access to time series for both the observed (and possibly aggregated) neural activities and external input. It is thus not limited to mesoscale models and could also be applied to, say, Hodgkin Huxley-type neurons in isolation or networks. Nevertheless, in this article, the underlying microscopic model does make the inferred parameters more readily interpretable and provides a good idea of what values an inference algorithm should find for the parameters.

Methods have recently been developed for inferring models where stochastic equations are treated as a black box simulator (Greenberg, Nonnenmacher, & Macke, 2019; Lueckmann et al., 2017; Papamakarios & Murray, 2016; Papamakarios, Sterratt, & Murray, 2018). In such a case, one does not have access to the internal variables of the model and thus cannot compute the likelihood of its parameters; instead, these methods make use of repeated simulations to find suitable parameters. While this makes them applicable to a wide range of models, the repeated simulations can make them computationally expensive and best suited to optimizing a set of statistical features rather than full-time traces. Moreover, for the models of interest here, the likelihood can be derived from the stochastic evolution equations.

We show in this work that the likelihood can indeed be used to infer model parameters using non-convex optimization. The resulting optimization problem shares many similarities with training recurrent neural networks (RNNs) popular in machine learning (Goodfellow, Bengio, & Courville, 2016; Waibel, Hanazawa, Hinton, Shikano, & Lang, 1989) and allows us to leverage optimization tools from that field. However, RNNs in machine learning are typically based on generic, nonmechanistic models, which implies that interpretation of the resulting network can be challenging (but see work on RNN visualization by Barak et al. (Barak, 2017; Haviv, Rivkind, & Barak, 2019; Sussillo & Barak, 2012). Thus, our approach can be regarded as complementary to RNN approaches, as we directly fit a mechanistically interpretable model.

This article is organized as follows. In sections 2.1 and 2.2, we establish that maximum likelihood inference for our chosen mesoscopic model is sound, and in section 2.3 we provide empirical estimates for the amount of data this procedure requires. Using the example of heterogeneous populations, section 2.4 then shows how inference can find effective parameters, which compensate for the mismatch between data and model. In section 2.5 we identify codependence between multiple model parameters by recovering the full Bayesian posterior. Finally, section 2.6 demonstrates that the approach scales well by considering a more challenging four-population model with 36 free parameters. Section 3 discusses our results, with an emphasis on circumscribing the class of models amenable to our approach. Method details are provided in section 4, along with technical insights gained as we adapted likelihood inference to a detailed dynamical model. Additional details, including a full specification of parameter values used throughout the article, are given in appendixes A to I.

2 Results

2.1 Model Summary. We studied the pair of microscopic and mesoscopic models presented in Schwalger et al. (2017), which is designed to represent excitatory (E) and inhibitory (I) populations of a putative cortical

column of four neural layers (Potjans & Diesmann, 2014). For this study, we considered only layers 2/3 and 4 and made minor parameter adjustments to maintain realistic firing rates (see appendix A). We also reduced all population sizes by a factor of 50 to ease the simulation of the microscopic model. This increases the variance of population activities and so does not artificially simplify the task of inferring mesoscopic parameters.

The microscopic model is composed of either two or four populations of generalized integrate-and-fire (GIF) neurons. Neurons are randomly connected, with connectivity probabilities depending on the populations. The combination of excitatory and inhibitory input, along with internal adaptation dynamics, produces for each neuron i a time-dependent firing rate $\lambda_i(t|\mathcal{H}_t)$; this rate is conditioned on the spike history up to t, denoted $\mathcal{H}_t$ (for equations, see section 4.1). Whether that neuron spikes within a time window $[t, t+\Delta t)$ is then determined by sampling a Bernoulli random variable (Schwalger et al., 2017),

$$s_i(t|\mathcal{H}_t) \sim \text{Bernoulli}(\lambda_i(t|\mathcal{H}_t)\Delta t), \tag{2.1}$$

where Δt is chosen such that $\lambda_i(t|\mathcal{H}_t)\Delta t \ll 1$ is always true; we later refer to this stochastic process as *escape noise*. If all parameters are shared across all neurons within each population, we call this a *homogeneous* microscopic model. Conversely, we call a model *heterogeneous* if at least one parameter is unique to each neuron. We denote $\mathcal{I}_\alpha$ the set of indices for neurons belonging to a population α.

The *expected activity* a_α of a population α is the normalized expected number of spikes,

$$a_\alpha(t|\mathcal{H}_t) = \frac{1}{N_\alpha}\sum_{i\in\mathcal{I}_\alpha}\lambda_i(t|\mathcal{H}_t), \tag{2.2}$$

which is a deterministic variable once we know the history up to t. In contrast, the *activity* A_α of that population is a random variable corresponding to the number of spikes actually observed:

$$A_\alpha(t|\mathcal{H}_t) := \frac{1}{N_\alpha}\sum_{i\in\mathcal{I}_\alpha}s_i(t|\mathcal{H}_t). \tag{2.3}$$

These variable definitions are summarized in Table 1.

In practice, data are discretized into discrete time steps $\{t_k\}_{k=1}^{L}$, which we assume to have uniform lengths Δt and to be short enough for spike events of different neurons to be independent within one time step (this condition is always fulfilled when the time step is less than the synaptic transmission delay). Under these assumptions, equation 2.3 can be approximated by a binomial distribution (Schwalger et al., 2017),

Table 1: Key Variable Definitions.

Variable	Definition
N_α	Number of neurons in population α
M	Number of populations, $\alpha = 1, \ldots, M$
L	Number of time steps used to compute the likelihood
Δt	Time step
$\mathcal{I}_\alpha$	Set of indices of neurons belonging to population α
$s_i(t)$	1 if neuron i spiked within time window $[t, t + \Delta t)$, 0 otherwise
$A_\alpha(t)$	Activity in population α averaged over time window $[t, t + \Delta t)$
$a_\alpha(t)$	Expectation of $A(t)$ conditioned on $\{A(t')\}_{t'<t}$

$$A_\alpha^{(k)} := A_\alpha(t_k|\mathcal{H}_{t_k}) \sim \frac{1}{N_\alpha \Delta t}\mathrm{Binom}(N_\alpha a_\alpha(t_k|\mathcal{H}_{t_k})\Delta t; N_\alpha). \tag{2.4}$$

If we repeat a simulation R times with the same input, we obtain an ensemble of histories $\{\mathcal{H}_{t_k}^r\}_{r=1}^R$ (due to the escape noise). Averaging over these histories yields the *trial-averaged activity*,

$$\bar{A}_\alpha^{(k)} := \frac{1}{R}\sum_{r=1}^{R} A_\alpha(t_k|\mathcal{H}_{t_k}^r), \tag{2.5}$$

the theoretical counterpart to the peristimulus time histogram (PSTH).

For the microscopic model, the history is the set of all spikes,

$$\mathcal{H}_{t_k} = \{s_i(t_l)\}_{\substack{i=1,\ldots,N \\ t_l<t_k}}. \tag{2.6}$$

To generate activities, we first generate spikes with equation 2.1 and use equation 2.3 to obtain activities (c.f. Figure 1A).

For the mesoscopic model, hereafter referred to as mesoGIF, the history contains only population activities:

$$\mathcal{H}_{t_k} = \{A_\alpha^{(l)}\}_{\substack{\alpha=1,\ldots,M \\ t_l<t_k}}. \tag{2.7}$$

The expected activity is then an expectation over all spike sequences consistent with that history, for which a closed-form expression was derived in Schwalger et al. (2017) (the relevant equations are given in appendix E). Activities are generated by using this expression to compute $a_\alpha(t)$ and then sampling equation 2.4. Unless mentioned otherwise, for the results reported in the following sections, we used the microscopic model for data generation and the mesoscopic model for inference.

In addition to homogeneity of populations and independence of spikes within a time step, the mesoscopic model depends on one more key approximation: that neuron populations can be treated as *quasi-renewal* (Naud & Gerstner, 2012; Schwalger et al., 2017). If neurons are viewed as having both refractory and adaptation dynamics, this is roughly equivalent to requiring that the latter be either slow or weak with respect to the former. (A typical example where this approximation does not hold is bursting neurons; see Naud & Gerstner, 2012.) Under these approximations, the unbounded history $\mathcal{H}_{t_k}$ can be replaced by a finite state vector $S^{(k)}$, which is updated along with the expected activity $a^{(t)}$ (see section 4.2). Since the update equations depend only on $S^{(k-1)}$, they are then Markovian in S. This in turn allows the probability of observations $P\left(A^{(L)}, A^{(L-1)}, \ldots, A^{(1)}\right)$ to be factorized as $P\left(A^{(L)}|S^{(L)}\right) \cdot P\left(A^{(L-1)}|S^{(L-1)}\right) \cdots P\left(A^{(1)}|S^{(1)}\right)$, which is key to making the inference problem tractable.

2.2 Recovering Population Model Parameters. We first consider a two-population model composed of E and I neurons. We use the homogeneous microscopic model to generate activity traces (see Figure 2A), with a frozen noise input, which is shared within populations; this input is sine-modulated to provide longer-term fluctuations (see equation 4.28). A maximum a posteriori (MAP) estimate $\hat{\eta}_{\mathrm{MAP}}$ of 14 model parameters is then obtained by performing stochastic gradient descent on the posterior (see section 4). Because the likelihood is nonconvex, we perform multiple fits, initializing each one by sampling from the prior (see Figure 2B). We then keep the one that achieves the highest likelihood, which in practice is often sufficient to find a near-global optimum (Meyer, Williamson, Linden, & Sahani, 2017).

An important note is that one can fit only parameters that are properly constrained by our data. For example, in the mesoGIF model, the firing probability is determined by the ratio (see equation 4.8)

$$\frac{u(t) - \vartheta(t)}{\Delta_{\mathrm{u}}}, \tag{2.8}$$

where u is the membrane potential, ϑ the firing threshold, and Δ_{u} a parameter describing the level of noise. All of these quantities are computed in units of millivolts, and the terms in the numerator depend on the resting potential u_{rest} and threshold u_{th}. However, since equation 2.8 is dimensionless, the choice of millivolts is arbitrary: after changing Δ_{u}, one can rescale u_{rest} and u_{th} (along with the synaptic weights w and reset potential u_{r}) to recover exactly the same dynamics. The set of parameters w, Δ_{u}, u_{rest}, u_{th} and u_{r} is thus degenerate, and they cannot all be inferred simultaneously; for this article, we set the voltage scale to millivolts by fixing u_{rest} and u_{th} to the values proposed by Schwalger et al. (2017). Other parameters are

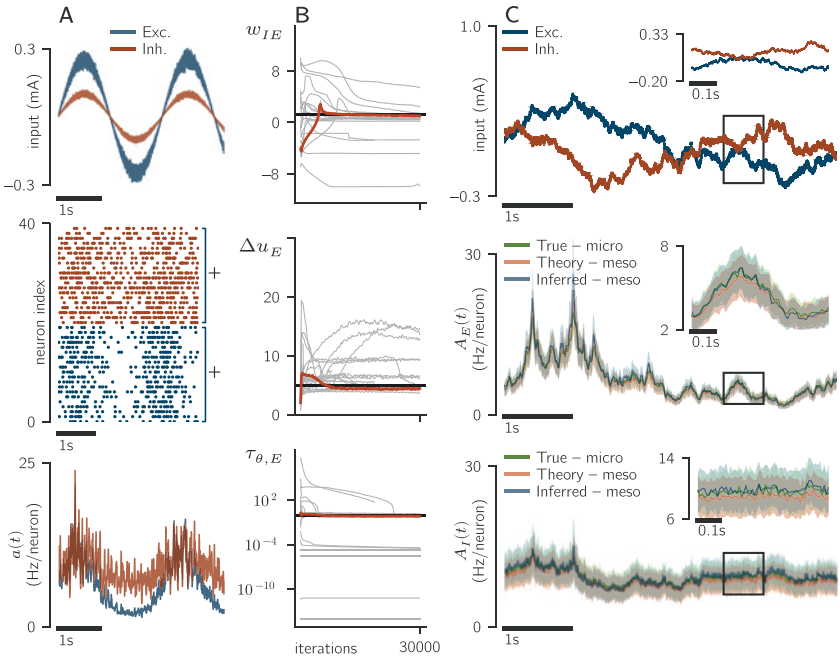

Figure 2: Inferred model generalizes to different inputs. (A) Data generation. Microscopic E and I populations receive a noisy sinusoidal input (see equation 4.28 and Table 5), which is shared across populations (top). Generated spikes (middle) are summed across each population, such that the inference algorithm sees only the total activity in each. Despite being deterministic given the history $\mathcal{H}$, the population-averaged expected activity (see equation 2.2) still shows substantial fluctuations due to stochasticity of the history itself (bottom). (B) Inference recovers parameter values close to those used to generate the data. We performed 25 fits, retaining the one that found the local optimum with the highest likelihood (shown in red). Black lines indicate the prediction of the mesoscopic theory of Schwalger et al. (2017), based on ground-truth values of the microscopic model. Fits for all 14 parameters are shown in Figure 12. (C) Inferred mesoscopic model reproduces input-driven variations in population activity. For testing, we used low-pass-filtered frozen white noise input (see equation 4.29, Table 6, top) to simulate the inferred mesoscopic model; middle and bottom plots respectively show the activity of the E and I populations. Each model was simulated 100 times; we show the mean and standard deviation over these realizations as lines and shading of corresponding colors. (Values were averaged over disjoint bins of 10 ms.) Performance measures are $\bar{\rho} = 0.950, 0.946, 0.918$ and $\mathrm{RMSE} = 3.42 \pm 0.07, 3.55 \pm 0.09, 3.4 \pm 0.08$ for the true, theory and inferred models, respectively (see section 4.7).

Table 2: Parameters for the Micro- and Mesoscopic Models.

Parameter	Number of Components: Two-Population Model	Number of Components: Four-Population Model	Description
p	4	16	Connection probability
$\boldsymbol{w}$	**4**	**16**	**Connection weight**
Δ	4	16	Transmission delay
N	2	4	Number of Neurons in population
R	2	4	Membrane resistance
u_{rest}	2	4	Membrane resting potential
$\boldsymbol{\tau_m}$	**2**	**4**	**Membrane time constant**
t_{ref}	2	4	Absolute refractory period
u_{th}	2	4	Nonadapting threshold
u_{r}	2	4	Reset potential
$\boldsymbol{c}$	**2**	**4**	**Escape rate at threshold**
$\boldsymbol{\Delta_u}$	**2**	**4**	**Noise level**
$\boldsymbol{\tau_s}$	**2**	**4**	**Synaptic time constant**
$\boldsymbol{J_\theta}$	**1**	**2**	**Adaptation strength**
$\boldsymbol{\tau_\theta}$	**1**	**2**	**Adaptation time constant**

Notes: For the mesoscopic populations, the ensemble of neuron parameters is replaced by a single effective value for that population. For each parameter, we indicate the number of components in the two- and four-population models; adaptation parameters have fewer components because the model assumes no adaptation for inhibitory neurons. Boldface is used to indicate inferred parameters; the remainder are fixed to the known ground truth values listed in Table 7. This results in, respectively, 14 and 36 free parameters for the two- and four-population models. A brief discussion of how we chose which parameters to infer is given at the end of section 4.5.

similarly ill constrained, and in total we inferred 14 model parameters; these are listed in Table 2.

We tested the inferred model on frozen low-pass-filtered white noise of the same form as in Augustin, Ladenbauer, Baumann, and Obermayer (2017); see Figure 2C, top), ensuring that a range of relevant time scales are tested. Despite the frozen input, variability between realizations does remain: for the GIF model this is due to sampling the escape noise (equation 2.1), while for the mesoGIF model, it is due to sampling the binomial in equation 2.4. We thus compare models based on the statistics of their response rather than single realizations: each model is simulated 100 times with different internal noise sequences (for each neuron in the case of the GIF model, and for each population in the case of the mesoGIF model) to produce an ensemble of realizations, from which we estimate the time-dependent mean and standard deviation of $A(t)$. Mean and standard deviation are then averaged over disjoint 10 ms windows to reduce variability due to the finite number of realizations. The results are reported as respectively, lines and shading in Figure 2C, and show agreement between true

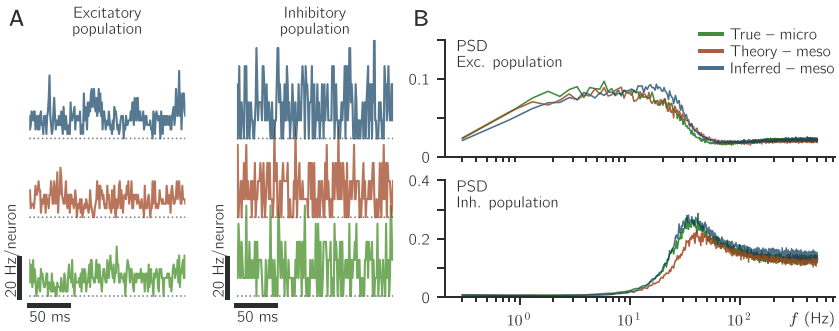

Figure 3: Inferred model reproduces expected power spectral density. (A) Segment of simulations of the same three models shown in Figure 2C under constant 0.5 mA input to both E and I populations. Dotted line indicates ordinate zero. (B) Power spectral density for the excitatory (top) and inhibitory (bottom) populations. For each model, spectra were computed for 50 distinct realizations of 9 s each and averaged. To reduce the error due to finite number of realizations, the frequency axis was then coarsened to steps of 0.5 Hz by averaging nonoverlapping bins.

and inferred models; we also find good agreement in the power spectrum of the response to constant input (see Figure 3). Parameterizations for the training and test inputs are given in section 4.8 and the full set of fits is shown in Figure 12.

2.3 Quantifying Data Requirements. While simulated data can be relatively cheap and easy to obtain, this is rarely the case of experimental data. An important question therefore is the amount required to infer the parameters of a model. To this end, we quantify in Figure 4 the accuracy of the inferred dynamics as a function of the amount of data.

In order to be certain our ground-truth parameters were exact, for this section we used the mesoGIF for both data generation and inference. This allows us to quantify the error on the inferred parameters, rather than just on the inferred dynamics. In a more realistic setting, data and model are not perfectly matched, and this will likely affect data requirements. Testing and training were done with different external inputs to avoid overfitting; as in section 2.2, we used a sinusoidal frozen white noise for training and a low-pass-filtered frozen white noise for testing. During training, E and I neurons had respective average firing rates of 5.9 and 8.4 Hz, which translates to approximately 3500 spikes per second for the whole population.

We measured the accuracy of inferred dynamics by simulating the model with both the ground truth and inferred parameters, generating 20 different realizations for each model. These were used to calculate both the per trial and trial-averaged Pearson correlation (ρ, $\bar{\rho}$) and root-mean-square

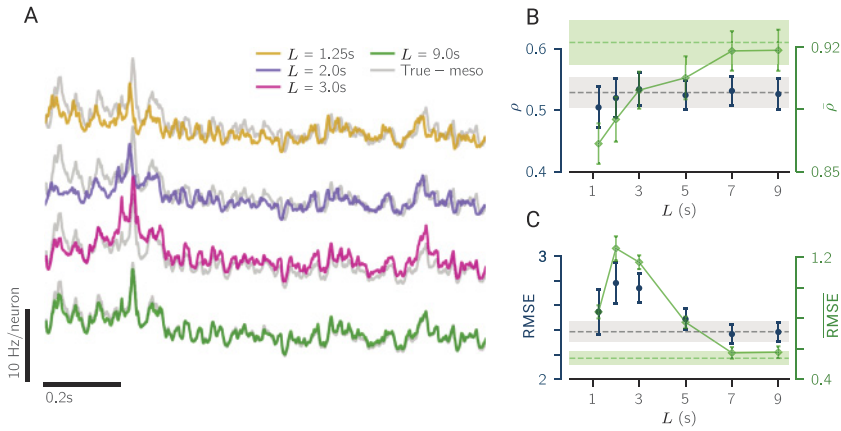

Figure 4: Inferred model no longer improves after 20,000 spikes. Model parameters were inferred from data generated using shared frozen noisy sinusoidal input and tested on low-pass-filtered frozen white noise. (A) Sample portion of the simulated traces used to compute discrepancy measures. Traces of the expected activity $a(t)$ of the excitatory population in a two-population E-I model, using parameters inferred from increasing amounts L of data; all simulations are done on test input using the same random seed to sample the binomial in equation 2.4. Note that the model did not see this input during training. (B, C) Inferrence performance, measured as either Pearson correlation ρ (B) or RMSE (C) between 20 simulations of the inferred and true mesoscopic models. Dashed lines indicate maximum achievable performance, estimated by computing the measures on a different set of 20 realizations of the ground-truth model; shading indicates standard deviation of that value. Blue points: per trial statistics (see equations 4.24 and 4.25); green points: trial-averaged traces (see equations 4.26 and 4.27). Trial-averaged errors were estimated by bootstrapping. Results suggests that performance is well summarized by $\bar{\rho}$ and RMSE.

error (RMSE, $\overline{\text{RMSE}}$) between models. An additional 20 simulations of the ground-truth model were used to estimate the best achievable performance for each measure. For per trial measures, the reported standard deviation provides an estimate of the variability between realizations; for trial-averaged measures, the standard deviation is obtained by bootstrapping, and is purely an uncertainty on the statistic (it vanishes in the limit of large number of realizations). The calculations for these measures are fully described in section 4.7. In subsequent section, we report only the values of $\bar{\rho}$, RMSE to avoid redundant information.

Consistent with the observations of Augustin et al. (2017), we found that ρ (in contrast to $\bar{\rho}$) does not allow differentiating between models close to

ground truth. The RMSE and $\overline{\text{RMSE}}$, on the other hand, showed similar sensitivity but may be unreliable far from ground truth (as evidenced by the data point at $L = 1.25$ s in Figure 4C). Since the per trial RMSE additionally quantifies the variability between realizations (through its standard deviation), we preferred it over its trial-averaged analog.

As we would expect, the inferred model better reproduces the dynamics of the true model when the amount of data is increased (see Figure 4); when fitting all 14 parameters of the mesoGIF model, the inferred model no longer improves when more than 5 s to 7 s of data are provided (see Figures 4B and 4C) – corresponding to a total of about 17,500 to 24,500 spikes. In appendix D, we repeat the test described here with smaller parameter sets (achieved by clamping certain parameters to their known ground-truth values). We find that this has only a modest effect on the achieved performance, but does significantly improve the consistency of fits (compare Figures 9B and 9C). Inferring larger parameter sets is thus expected to require more fits (and consequent computation time) before a few of them find the MAP. Certain parameters are also more difficult to infer: for the case shown in Figure 4, relative errors on the inferred parameters range from 5% to 22% (see appendix D, Table 12). Parameters describing the inhibitory population ($\tau_{m,I}$, w_{IE}, w_{II}) show the highest relative error, as well as the escape rates (c_E, c_I) and the adaptation time constant ($\tau_{\theta,E}$).

2.4 Modeling High-Dimensional Heterogeneous Populations with an Effective Low-Dimensional Homogeneous Model. A frequently understated challenge of meso- and macroscale population models is that of choosing their parameters such that the dynamics of the modeled neuron populations are consistent with the high-dimensional dynamics of networks of individual neurons. A typical approach, when measurements of microscopic single-neuron parameters are available, is to assign each parameter its mean across the population (Gerstner, Paninski, Naud, & Kistler, 2014, sec. 12). However, as alluded to in section 1, mean parameters do not always make good predictors for nonlinear systems; this is evidenced by Figure 5, which expands on Figure 1B.

An alternative approach would be to fit the population model to observed population activities, such as to ensure maximum consistency with data—for example, by finding the maximum a posteriori (MAP) parameters. In this way, we obtain effective parameters that compensate for the mismatch between data and population model.

To show that this can work, we made the microscopic model heterogeneous in three parameters: $\tau_{m,E}$, $\tau_{m,I}$, and $\tau_{\theta,E}$. These parameters were set individually for each neuron by sampling from a log-normal distribution (see Figure 5A and Table 8). As in previous sections, output from the microscopic model under sine-modulated frozen white noise input was then used to train the mesoscopic one. For testing, we used a single step input

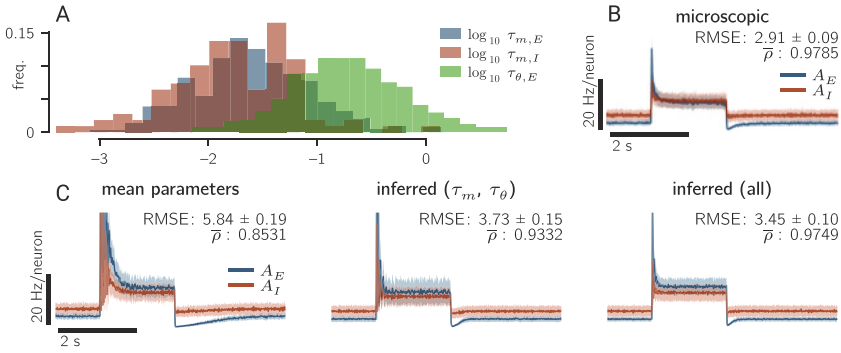

Figure 5: Inferred effective parameters can compensate for mismatch between microscopic and mesoscopic models. (A) A heterogeneous microscopic model of two populations was constructed by sampling three time constants from log-normal distributions (see Table 8). All other parameters are as in section 2.2 and Figure 2, and the same sine-modulated white noise as in Figure 2A was used to train the model. (B) Heterogeneous microscopic model driven by a single step current. Shown are the mean (line) and standard deviation (shading) of the model's response, computed from 60 realizations and averaged over disjoint windows of 10 ms. Realizations differ due to sampling the escape noise. (C) Simulations of the mesoscopic model with the same step input as in panel B, using mean parameters (left), inferred τ_m and τ_θ (middle, with all other parameters homogeneous and set to ground truth), and the inferred full 14-parameter set (right). Line and shading have the same meaning as in panel B) and are based on 50 realizations for each model; these differ by the sampling of the binomial in equation 2.4. We see that inferred models more closely reproduce the trace in panel B, which is confirmed by the decreased RMSE and increased $\bar{\rho}$.

(see Figure 5B); this allowed us to test the performance of the inferred model in both the transient and steady-state regimes. The per trial RMSE and trial-averaged correlation $\bar{\rho}$ were computed on ensembles of realizations, as described in section 4.7.

We considered three sets of parameters for the mesoscopic model. For the first, we set $\tau_{m,E}$, $\tau_{m,I}$, and $\tau_{\theta,E}$ to their sample averages. This produced rather poor results (see Figure 5C, left); in particular, the transient response to the step is much more dramatic and long-lived than that of the ground-truth model. As the neural model is highly nonlinear in its parameters, linearly averaging parameters is not guaranteed to produce optimal results.

The test results are improved when the heterogeneous parameters are inferred (see Figure 5C, middle). However, fitting only the heterogeneous parameters gives the mesoscopic model only three degrees of freedom to compensate for approximating a heterogeneous model by a homogeneous one, and it still produces traces with a variance that is too high. Indeed,

giving the model full freedom over the parameters provides another step improvement (see Figure 5C, right), with output from the mesoscopic model differing from the target output only by a higher transient peak and slightly different mean activities (obtained parameter values are listed in Table 9). Thus while fitting more parameters may incur additional computational cost (see appendix D), it also provides more opportunities to accommodate model mismatch.

The results of this section show the necessity of inferring population parameters rather than simply averaging single-neuron values. It also demonstrates the ability of population models to reproduce realistic activities when we provide them with good effective parameters; in order to compensate for modeling assumptions, those parameters will in general differ from those of a more detailed microscopic model.

2.5 Full Posterior Estimation over Parameters. It can often be desirable to know which parameters, or combinations of parameters, are constrained by the data. Bayesian inference, that is, estimation of the posterior distribution over parameters given the data, can be used to not only identify the best-fitting parameters but also to characterize the uncertainty about these estimates. Notably, these uncertainties may be highly correlated across parameters. For instance, one expects an increase in E connectivity to cancel a decrease in (negative) I connectivity to the same population, and this is confirmed by the correlation in the marginals shown in Figure 6A. Interestingly, this correlation is in fact stronger for connectivities sharing the same *target* than those sharing the same *source*. More novel structure can be learned from Figure 6B, such as the strong correlation between the adaptation parameters or the complete absence of correlation between them and the synaptic parameters. In particular, the tight relationship between $J_{\theta,E}$, $\tau_{\theta,E}$, and c_E suggests that for determining model dynamics, the ratios $J_{\theta,E}/\tau_{\theta,E}$ and $J_{\theta,E}/c_E$ may be more important than any of those three quantities individually.

Since there are 14 unknown parameters, the posterior is also 14-dimensional; we represent it by displaying the joint distributions between pairs, obtained by marginalizing out the other 12 parameters (see section 4.6). Training data here were generated in the same way as in section 2.2, from a homogeneous microscopic model with the parameters listed in Table 2. To provide a sense of scale, we have drawn ellipses in Figure 6 to indicate the volume corresponding to two standard deviations from the mean under a gaussian model. In a number of cases, it highlights how the true distribution is nongaussian—for example, the distributions of c_E, $J_{\theta,E}$, and $\tau_{\theta,E}$ are noticeably skewed.

A naive way to compute these 2D marginals would be to numerically integrate the likelihood; however, given that that leaves 12 dimensions to integrate, such an approach would be computationally unfeasible. Instead we

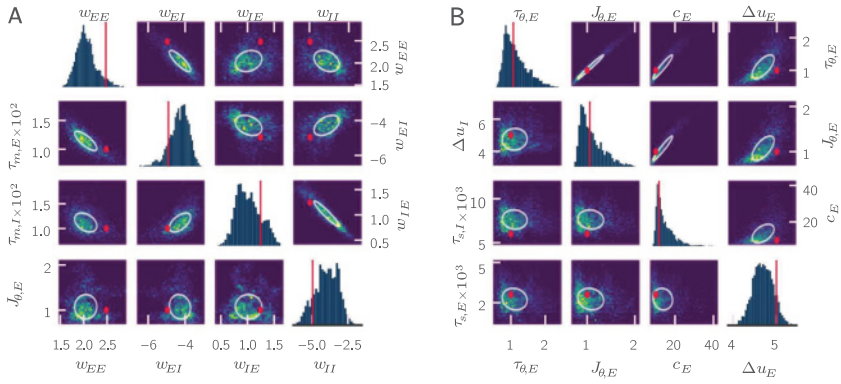

Figure 6: Posterior probability highlights dependencies between model parameters. Panels show one- and two-parameter marginals; all panels within a column use the same parameter for their abscissa. (A) Above diagonal: Full posterior over the connectivities w. Strongest (anti)correlation is between pairs impinging on the same population (i.e., w_{EI}–w_{EE} and w_{IE}–w_{II}.) Below diagonal: Membrane time constants and adaptation strength show correlations with connectivity. Panels on the diagonal show the marginal for that column's parameters. Red dot or line shows the parameters' ground-truth values. The ellipse is centered on the mean and corresponds to two standard deviations under a gaussian model. The full posterior over all 14 parameters is shown in Figure 11 and was obtained with HMC sampling using data generated with the two-population homogeneous microscopic model. (B) Above diagonal: Tight correlation between $\tau_{\theta,E}$, $J_{\theta,E}$, and c_E suggests their ratios are most important to determining model dynamics. Below diagonal: There is little correlation between adaptation and synaptic parameters. Diagonal panels, red marks, and ellipses are as in panel A.

used Hamiltonian Monte Carlo (HMC) sampling (Betancourt & Girolami, 2013; Neal, 2012). Monte Carlo methods are guaranteed to asymptotically converge to the true posterior, a valuable feature when one wishes to deduce interactions between parameters from its structure. Nevertheless, due to the complexity of mesoGIF's likelihood, memory and computational cost still required special consideration (see section 4.6).

We note that the 2σ ellipses in Figure 6, while informative, are imperfect indicators of the probability mass distribution. If the posterior is gaussian, then each projection to a 2D marginal places 86.5% of the probability mass within the ellipse; however, for nongaussian posteriors, this number can vary substantially. Moreover, the markers for ground-truth parameters shown in Figure 6 may differ from the effective parameters found by the model (see section 2.4).

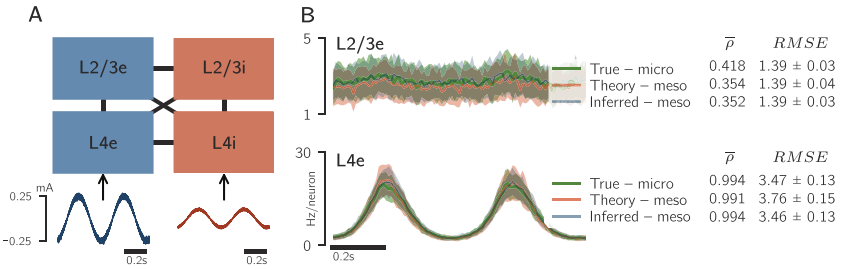

Figure 7: Inference of a four-population model with 36 free parameters. (A) Model represented E (blue) and I (red) populations from layers L2/3 and L4 of a cortical column. During training, only L4 populations received external sinusoidal input. The homogeneous microscopic model was used to generate data. (B) The mesoscopic model matches aggregate microscopic dynamics ("True – micro"), both when using theoretical ("Theory – meso") and inferred parameters ("Inferred – meso"). In contrast to the previous section, correlation and RMSE scores are reported separately for each population; they are computed from 60 realizations of each models.

2.6 Pushing the Limits of Generalization. The previous sections have shown that we can recover 14 parameters of the two population mesoGIF model. A natural question is whether this approach scales well to larger models. We investigated this by considering four neuron populations representing the L2/3 and L4 layers of the Potjans-Diesmann microcircuit (Potjans & Diesmann, 2014). The associated higher-dimensional set of mesoscopic equations follows the same form as in previous sections (Schwalger et al., 2017). There are 36 free parameters in this model, of which 16 are connectivities; they are listed in Table 2. Similar to previous sections, we trained mesoGIF on output from the microscopic model with sinusoidal drive (see Figure 7A).

The L4 populations tend to drive the activity in this model, and we found that we do not need to provide any input to the L2/3 neurons to get parameter estimates that accurately predict population activity (see Figure 8, left): the small fluctuations in L2/3 (see Figure 7B) suffice to provide constraints on those population parameters. Those constraints, of course, are somewhat looser, and in particular connection strengths onto L4 are not as well estimated when compared to ground truth (see Table 10).

Pushing the mesoscopic approximation beyond its validity limits using inputs with abrupt transitions understandably increases the discrepancy between ground truth and model (see Figure 8, right). Indeed, such a strong input may cause neurons to fire in bursts, thereby breaking the quasi-renewal approximation (see section 2.1). During an input spike, the true model shows small oscillations; the theoretical mesoGIF reproduces these oscillations but with an exaggerated amplitude and higher variance

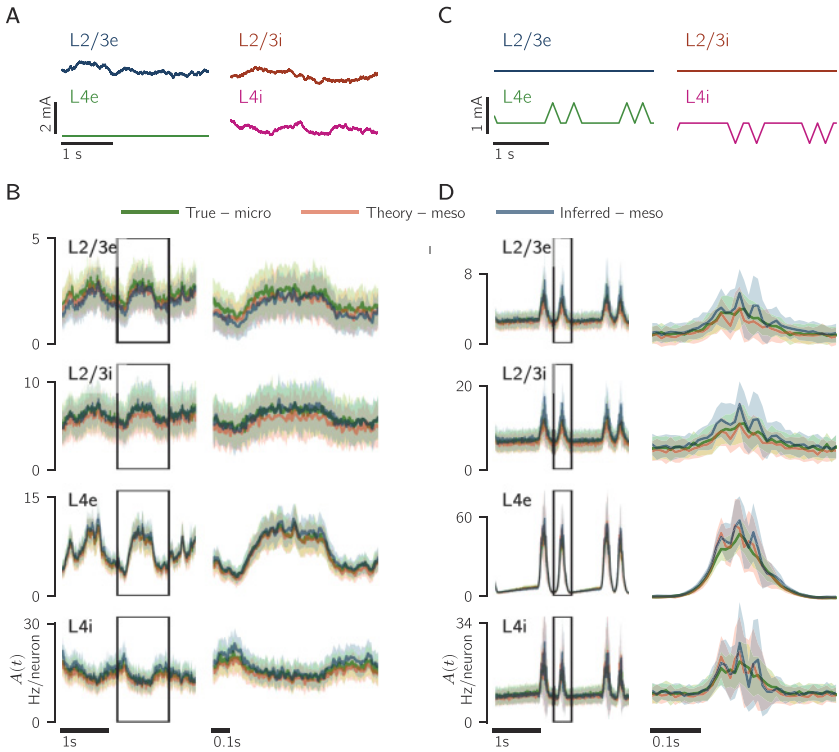

Figure 8: Generalization errors appear with large deviations from the training input. We test the 36 parameter model inferred in Figure 7 under two different stimulation protocols. Lines and shading show mean and standard deviation over 60 realizations, computed as in section 2.2. (A, B) After completely removing external inputs to L4e (compare A with the training input in Figure 7A), predictions of the inferred and theoretical models are still indistinguishable. (C, D) To obtain visible deviations between inferred and theoretical models, we used inputs (C) that stretch the mesoGIF assumptions. Oscillations are present in both the microscopic and mesoscopic models, but in the latter they have much larger amplitudes: compare the blue and red traces to the thicker green trace in panel D.

between realizations, and in contrast to section 2.4, the inferred model does no better. This larger discrepancy with the true model is reflected in the performance measures (see Tables 14 and 15), and is consistent with the observation that the mesoGIF has higher variance during bursts (Schwalger et al., 2017, p. 15). Slower timescale dynamics are still accurately captured by both the theoretical and inferred models.

The capacity of the inferred model to generalize to unseen inputs is thus quite robust, with discrepancies between inferred and ground-truth models occurring only when the test and training input were very different. Of course this is in part due to mesoGIF being a good representation of the activity of homogeneous GIF neurons: while inference may compensate for some discrepancies between the model and the data, it still can work only within the freedom afforded by the model.

3 Discussion

Population models play a key role in neuroscience: they may describe experimental data at the scale they are recorded and serve to simplify the dynamics of large numbers of neurons into a human-understandable form. These dynamics may occur on a range of scales, from the mesoscopic, limited to a single cortical column, to the macroscopic, describing interactions between regions across the entire brain. Mechanistic models allow us to bridge those scales, relating microscale interactions to meso- or macroscale dynamics; of these, the model chosen for this study allows for rich dynamics at the single level by including synaptic, refractory, and adaptation dynamics.

We have demonstrated that it is possible to fit a mechanistic population model to simulated data by maximizing the likelihood of its parameters, in much the same way as is already done with phenomenological models (Macke et al., 2011; Pillow et al., 2008; Zhao & Park, 2016). Since mechanistic models describe concrete, albeit idealized, biophysical processes, they have the additional benefit that their parameters can be understood in terms of those processes. Moreover, those parameters are typically not dependent on the applied input, and thus we can expect the inferred model to generalize to novel stimulus conditions.

We also found that after making a few parameters heterogeneous, averaging did not recover the most representative parameters. In general, when there is discrepancy between model and data, the effective parameters are difficult to recover analytically; data-driven methods then provide a valuable supplement to theoretical analysis in order to ensure that a model actually represents the intended biological process. Nevertheless, since the inference procedure is agnostic to the model, it is up to the modeler to choose one for which the effective parameters remain interpretable.

The approach we have presented requires only that a differentiable likelihood function be available and thus is not limited to neuron population models. Stochastic models of neuron membrane potentials (Goldwyn & Shea-Brown, 2011), animal populations (Wood, 2010), and transition phenomena in physics and chemistry (Horsthemke & Lefever, 2006) are examples for which parameters could be inferred using this approach.

In practice we expect some models to be more challenging than others. For instance, evaluating the likelihood of a spiking model typically

involves integrating over all time courses of the subthreshold membrane potential compatible with the observed spike train (Paninski, Pillow, & Simoncelli, 2004). This integral can be difficult to evaluate accurately, especially for models incorporating adaptation and refractoriness (Mena & Paninski, 2014; Ramirez & Paninski, 2014). If evaluation of the likelihood is prohibitively expensive, likelihood-free approaches might be more appropriate (Lueckmann et al., 2017; Papamakarios & Murray, 2016).

Also of note is that we required the dynamics to be formulated as a Markov process to express the likelihood (see section 4.3). We achieved this by constructing a state vector, but the size of this vector adds substantial computational cost, and in practice there is a trade-off between the length of the integration time window and the number of units (here, neuron populations) we can infer. Since neural field models are also computationally represented by long state vectors, inference on these models would be subject to a similar trade-off. Finally, our current implementation assumes that the state S (see section 4.2) can be fully reconstructed from observations. If only a partial reconstruction of S is possible, undetermined components of S form a latent state that must be inferred along with the parameters. This type of problem has already been studied in the context of dimensionality reduction (Cunningham & Yu, 2014; Macke et al., 2011; Rule, Schnoerr, Hennig, & Sanguinetti, 2019), and it is conceivable that such methods could be adapted to our framework. Such an approach would allow one to perform dimensionality reduction with mechanistic models of temporal dynamics.

The work of Rule et al. (2019) presents an interesting complement to ours. The authors consider a neural field model where activities are observed only indirectly via a point process, thus addressing the problem of inferring latent states. They infer both these states and the point-process parameters but assume known parameters and neglect finite-size effects for the mesoscopic model; in contrast, here we inferred the mesoscopic model parameters while assuming that population states are observed. Inferring both mesoscopic model parameters and latent states remains a challenge for both of these approaches.

To obtain posteriors, we employed a Hamiltonian Monte Carlo algorithm with minimal automatic tuning. We found this to work better than a more automatically tuned variant (see section 4.6), but it is beyond the scope of this work to provide a complete survey of sampling methods. The applicability of more recently developed algorithms such as Riemann manifold Monte Carlo (Girolami & Calderhead, 2011), sequential Monte Carlo (Moral, Doucet, & Jasra, 2006), and nested sampling (Skilling, 2006) would be worth exploring in future work. Variational methods such as that described by Kucukelbir, Tran, Ranganath, Gelman, and Blei (2017) are another alternative to estimating posteriors that do not require sampling at all. They generally scale to large parameter spaces but do not provide the asymptotic guarantees of MCMC and may artificially smooth the resulting posterior.

Important obstacles to using inference on complex models are the implementation and computational costs. Software tools developed for this work have helped limit the former, but the latter remain a challenge, with many of the figures shown requiring multiple days of computation on a personal workstation. While manageable for studying fixed networks, this would become an impediment for scaling to larger models or tracking the evolution of parameter values by inferring them on successive time windows. For such tasks, further work would be required to reduce the inference time, for example, by investigating how large the integration time step for the mesoGIF model can be made or by optimizing the current implementation. One might also attempt to derive a computationally simpler model or make better use of parallelization and/or graphical processing units.

As Rule et al. (2019) noted, an additional complication to inferring mechanistic model parameters is that they may be underconstrained. In our case, since mesoGIF is a rate model, the voltage scale can be chosen freely by setting the resting (u_{rest}) and threshold (u_{th}) potentials; if we nonetheless attempt to infer them along with the noise scale (Δ_u), fits are unable to converge (see sections 2.2 and 4.5). We avoided this problem by identifying the problematic parameters and fixing them to their known values. However, the development of a more systematic approach to dealing with underconstrained parameters is left for future investigations.

Since inference time is highly dependent on computational complexity, there is a trade-off between bottom-up models that attempt to match dynamics as closely as possible and simpler top-down models that aim for computational efficiency; while the latter tend to provide better scalability, the former are likely to be more interpretable and allow for extrapolation to new dynamical regimes (see section 2.6). Choosing the right model thus remains a key component of data analysis and modeling.

Inference methods based on machine learning allow for flexible model design, using known biophysical parameter values when they are available and inference to determine the others that are consistent with data. We hope this work further motivates the use of richer models in neuroscience by providing tools to fit and validate them.

4 Methods

4.1 Microscopic Model. We consider an ensemble of neurons grouped into M populations; the symbols i, j are used to label neurons and α, β to label populations. The neuron indices i, j run across populations and are thus unique to each neuron.

Each neuron i produces a spike train represented as a sum of Dirac delta functions,

$$s_i(t) = \sum_k \delta(t - t_{i,k}), \tag{4.1}$$

where $t_{i,k}$ is the time of its kth spike. We denote Γ_i^β the set of neuron indices from population β that are presynaptic to neuron i, $w_{\alpha\beta}$ the strength of the connection from a neuron in population β to another in population α, and $\Delta_{\alpha\beta}$ the transmission delay between the two populations. As in Schwalger et al. (2017), we assume that intrinsic neural parameters are homogeneous across a given population. We further assume that connection strengths depend only on the source and target populations; for a connection between neurons of population β to those of population α, the strength is either $w_{\alpha\beta}$ with probability $p_{\alpha\beta}$ or zero with probability $1 - p_{\alpha\beta}$. Each spike elicits a postsynaptic current, which we sum linearly to obtain the synaptic inputs to neuron i from M populations,

$$R_\alpha I_{\mathrm{syn},i}(t) = \tau_m^\alpha \sum_{\beta=1}^{M} w_{\alpha\beta} \sum_{j\in\Gamma_i^\beta} \big(\epsilon_{\alpha\beta} * s_j\big)(t). \tag{4.2}$$

The transmission delay is captured by shifting the synaptic kernel with a Heaviside function Θ:

$$\epsilon_{\alpha\beta}(t) = \Theta(t - \Delta_{\alpha\beta}) \frac{e^{-(t-\Delta)/\tau_{s,\beta}}}{\tau_{s,\beta}}. \tag{4.3}$$

Spike generation is modeled by a *generalized integrate-and-fire* mechanism: leaky integration with adapting threshold, followed by an escape rate process. For each neuron i, the membrane potential u_i and firing threshold ϑ_i evolve according to

$$\tau_{m,\alpha} \frac{du_i}{dt} = -u_i + u_{\mathrm{rest},\alpha} + R_\alpha I_{\mathrm{ext},\alpha}(t) + R_\alpha I_{\mathrm{syn},i}(t); \tag{4.4}$$

$$\vartheta_i(t) = u_{\mathrm{th},\alpha} + \int_{-\infty}^{t} \theta_\alpha(t - t') s_i(t')\, \mathrm{d}t'. \tag{4.5}$$

Here, θ_α is the adaptation kernel for population α and $I_{\mathrm{ext},\alpha}$ the external input to that population. For this work, we used an exponential adaptation kernel,

$$\theta_\alpha(t) = \frac{J_{\theta,\alpha}}{\tau_{\theta,\alpha}} e^{-t/\tau_\theta}, \tag{4.6}$$

which allows us to rewrite equation 4.5 as

$$\tau_\theta \frac{d\vartheta_i}{dt}(t) = -\vartheta_i(t) + u_{\mathrm{th},\alpha} + J_{\theta,\alpha} s_i(t). \tag{4.7}$$

Spikes are generated stochastically with an escape rate (also called conditional intensity or hazard rate), calculated with the inverse link function f:

$$\lambda_i(t) = f(u_i(t) - \vartheta_i(t)). \tag{4.8}$$

For this work, we used

$$\lambda_i(t) = c_\alpha \exp((u_i(t) - \vartheta_i(t))/\Delta_{\mathrm{u},\alpha}), \tag{4.9}$$

where Δ_u parameterizes the amount of noise (or, equivalently, the softness of the threshold) and c is the firing rate when $u(t) = \vartheta(t)$.

Once a spike is emitted, a neuron's potential is reset to u_r and clamped to this value for a time t_ref corresponding to its absolute refractory period. It then evolves again according to equation 4.4. All model parameters are summarized in Table 2.

4.2 Mesoscopic Model. The mesoscopic equations describe the interaction of population activities (total number of spikes per second per neuron) in closed form: they can be integrated without the need to simulate individual neurons. This is achieved by identifying each neuron i by its age τ_i and making the assumptions stated in section 2.1: that each population is homogeneous, that neurons are all-to-all connected with effective weights $p_{\alpha\beta}w_{\alpha\beta}$, and that dynamics are well approximated as a quasi-renewal process. Under these conditions, it is possible to rewrite the dynamical equations in terms of the refractory densities $\rho_\alpha(t, \tau)$: the proportion of neurons with age $\tau_i \in [\tau, \tau + d\tau)$ in each population α. With very large populations N_α, we can neglect finite-size fluctuations and ρ satisfies the transport equation (Chizhov & Graham, 2008; Gerstner, 2000; Gerstner et al., 2014; Wilson & Cowan, 1972):

$$\frac{\partial \rho_\alpha}{\partial t} + \frac{\partial \rho_\alpha}{\partial \tau} = -\lambda_\alpha(t, \tau)\rho, \qquad \rho_\alpha(0, t) = A_\alpha(t). \tag{4.10}$$

Neuronal dynamics and synaptic interactions are captured within the functional form of the hazard rate $\lambda_\alpha(t, \tau)$, which depends only on τ and on the history of population activities. In the limit $N_\alpha \to \infty$, the evolution of $A(t)$ matches its expectation $a(t)$ and is obtained by integrating over all neurons:

$$N_\alpha \to \infty : \quad A_\alpha(t) = a_\alpha(t) = \int_0^\infty \lambda_\alpha(t, \tau)\rho_\alpha(t, \tau)\,\mathrm{d}\tau. \tag{4.11}$$

For finite N, the expression for the expected activity becomes (Schwalger & Chizhov, 2019; Schwalger et al., 2017)

$$a_\alpha(t) = \int_0^\infty \lambda_\alpha(t,\tau)\rho_\alpha(t,\tau)d\tau + \Lambda_\alpha(t)\left(1 - \int_0^\infty \rho_\alpha(t,\tau)\right), \tag{4.12}$$

where $\Lambda(t)$ is a rate function that accounts for finite-size effects in the refractory density. The activity then follows a stochastic process described by

$$A_\alpha(t) = \frac{n_\alpha(t)}{Ndt}, \qquad n_\alpha(t) \sim \mathrm{Binom}(Na(t)dt; N_\alpha). \tag{4.13}$$

For this work, we discretize time into steps of length Δt, and instead of the refractory density work with the vector $m_\alpha^{(k)}$, where $m_{\alpha,l}^{(k)}$ is formally defined as the expected number of neurons of age $\tau \in [l\Delta t, (l+1))$:

$$m_{\alpha,l}^{(k)} = \int_\tau^{\tau+\Delta t^-} N_\alpha \rho_\alpha(t_k, l\,\Delta t)\, d\tau, \qquad (l = 1, \ldots, K < \infty). \tag{4.14}$$

Here the superscript (k) indicates the simulation time step and l the age bin. Since refractory effects are negligible for sufficiently old neurons, $m^{(k)}$ only needs to be computed for a finite number of age bins K (see appendix E, as well as equation 86 from Schwalger et al., 2017).

We similarly compute the firing rates at time t_k as a vector $\lambda_{\alpha,l}^{(k)}$, $l = 1, \ldots, K$. The expected number of spikes in a time bin,

$$\bar{n}_\alpha^{(k)} = \mathrm{E}\left[n_\alpha^{(k)}\right], \tag{4.15}$$

can then be computed in analogy with equation 4.12, by summing the products $\lambda_{\alpha,l}^{(k)} m_{\alpha,l}^{(k)}$ over l and adding a finite-size correction; the precise equations used to evaluate $m_{\alpha,l}^{(k)}$, $\lambda_{\alpha,l}^{(k)}$, and $\bar{n}_\alpha^{(k)}$ are listed in appendix E. We can convert spike counts to activities by dividing by $N_\alpha \Delta t$:

$$a_\alpha^{(k)} := \frac{\bar{n}_\alpha^{(k)}}{N_\alpha \Delta t}, \qquad A_\alpha^{(k)} := \frac{n_\alpha^{(k)}}{N_\alpha \Delta t}. \tag{4.16}$$

For the following, it will be convenient to define the single-neuron firing probability,

$$p_{\alpha,\eta}^{(k)} := \frac{\bar{n}_{\alpha,\eta}^{(k)}}{N_\alpha}, \tag{4.17}$$

where the subscript η makes explicit the dependence on the model parameters. This allows us to rewrite equation 2.4 as

$$n_\alpha^{(k)} \sim \text{Binom}\left(p_{\alpha,\eta}^{(k)}; N_\alpha\right), \tag{4.18}$$

where $p_{\alpha,\eta}^{(k)} = p_{\alpha,\eta}(t_k|\mathcal{H}_{t_k})$ depends on the activity history $\mathcal{H}_{t_k}$ (see equation 2.7). Because K is finite, we can replace $\mathcal{H}_{t_k}$ by a finite state-vector $S^{(k)}$, obtained by concatenating all variables required to update $n^{(k)}$ (see appendix E, especially equation E.1):

$$S^{(k)} = \left(n^{(k)}, m^{(k)}, \lambda^{(k)}, \dots\right). \tag{4.19}$$

The update equations for $S^{(k)}$ are Markovian by construction, which simplifies the expression of the model's likelihood presented in the next section.

4.3 Likelihood for the Mesoscopic Model. As stated in section 4.2, the mesoGIF model can be cast in a Markovian form, which allows us to expand the probability of observing a sequence of spike counts as a recursive product. If that sequence has length L and an initial time point k_0, then that probability is

$$p\left(\left\{n_\alpha^{(k)}\right\}_{\substack{k=k_0\dots k_0+L-1\\ \alpha=1\dots M}}\right) = \prod_{\alpha=1}^{M}\prod_{k=k_0}^{L+k_0-1} p\left(n_\alpha^{(k)}|S^{(k)}\right). \tag{4.20}$$

The likelihood of this sequence then follows directly from the probability mass function of a binomial, using the definitions for $n_\alpha^{(k)}$ and $p_{\alpha,\eta}^{(k)}$ defined above:

$$L_{k_0;L} = \prod_{\alpha=1}^{M}\prod_{k=k_0}^{k_0+L-1} \binom{N_\alpha}{n_\alpha^{(k)}} \left(p_{\alpha,\eta}^{(k)}\right)^{n_\alpha^{(k)}} \left(1-p_{\alpha,\eta}^{(k)}\right)^{N_\alpha - n_\alpha^{(k)}}. \tag{4.21}$$

We note that the $n_\alpha^{(k)}$ are observed data points and are thus constant when maximizing the likelihood.

Expanding the binomial coefficient, the log likelihood becomes

$$\begin{aligned}\log L_{k_0;L}(\eta) = \sum_{\alpha=1}^{M}\sum_{k=k_0}^{k_0+L-1} &\log\left(N_\alpha!\right) - \log\left(n_\alpha^{(k)}!\right) - \log\left(\left(N_\alpha - n_\alpha^{(k)}\right)!\right) \\ &+ n_\alpha^{(k)}\log\left(\tilde{p}_{\alpha,\eta}^{(k)}\right) + \left(N_\alpha - n_\alpha^{(k)}\right)\log\left(1-\tilde{p}_{\alpha,\eta}^{(k)}\right),\end{aligned} \tag{4.22}$$

where we clipped the probability $\tilde{p}_\alpha^{(k)}$ to avoid writing separate expressions for $p_{\alpha,\eta}^{(k)} \in 0, 1$,

Algorithm 1: Silent initialization scheme.

1: $n_\alpha \leftarrow 0$
2: $h_\alpha, u_{\alpha,i} \leftarrow u_{\mathrm{rest},i}$
3: $x_\alpha \leftarrow N_\alpha$
4: $\lambda_{\alpha,i}, \lambda_{\mathrm{free},\alpha}, g_\alpha, m_{\alpha,i}, v_{\alpha,i}, y_{\alpha\beta}, z_\alpha \leftarrow 0$

$$\tilde{p}^{(k)}_{\alpha,\eta} = \begin{cases} \epsilon & \text{if } p^{(k)}_{\alpha,\eta} \leq \epsilon, \\ p^{(k)}_\alpha & \text{if } \epsilon \leq p^{(k)}_{\alpha,\eta} \leq 1-\epsilon, \\ 1-\epsilon & \text{if } p^{(k)}_{\alpha,\eta} \geq 1-\epsilon. \end{cases} \tag{4.23}$$

Clipping also avoids issues where the firing probability $p^{(k)}_\alpha$ exceeds one, which occurs when one explores the parameter space. (This can happen when parameters are such that the chosen Δt is no longer small enough for the underlying Poisson assumption to be valid, although it should *not* occur around the true parameters. (See the discussion by Schwalger et al., 2017, p. 48.) We found that with double precision, a tolerance $\epsilon = 1 \times 10^{-8}$ worked well.

For numerical stability, logarithms of factorials are computed with a dedicated function such as SciPy's `gammaln` (Jones, Oliphant, & Peterson, 2001). For optimization, the term $\log(N_\alpha!)$ can be omitted from the sum since it is constant.

4.4 Initializing the Model. Although the updates to the state S are deterministic (see section 4.2), only the components $n^{(k_0)}_\alpha$ of the initial state $S^{(k_0)}$ are known; unobserved components can easily number in the thousands. We get around this problem in the same manner as in Schwalger et al. (2017): by making an initial guess that is consistent with model assumptions (e.g., survival counts sum to N_α) and letting the system evolve until it has forgotten its initial condition. We note that the same problem is encountered when training recurrent neural networks, whereby the first data points are used to burn-in unit activations before training can begin. For the results we presented, we used a variation of the initialization scheme used by Schwalger et al. (2017), which we call the "silent initialization." In this scheme, neurons are assumed to have never fired, and thus they are all "free" (see algorithm 1). This results in large spiking activity in the first few time bins, which then relaxes to realistic levels.

This initialization scheme has the advantage of being simple and needing no extra computation, but with the high-dimensional internal state S, it also requires a large burn-in time of around 10 s. This can be largely mitigated by using sequential batches (see algorithm 2).

Algorithm 2: Training with sequential mini-batches. The gradient is normalized before computing Adam updates. Note that the state is not reinitialized within the inner loop.

repeat
  $S \leftarrow$ initialize state
  $k' \sim \mathrm{Uniform}(0, \gamma_L B)$ ▷ Randomize initialization burn-in
  $k_0 \leftarrow L_{\mathrm{burnin}} + k'$
  while $k_0 < L - B$ **do** ▷ Scan data sequentially
    $g \leftarrow \nabla \log L(\eta, A_{k_0:k_0+B})$ ▷ Log-likelihood gradient on the mini-batch
    if any$(|g| > g_{\mathrm{clip}})$ **then** ▷ Normalize gradients with L^∞ norm
      $g_{\max} \leftarrow \max(|\Delta\eta|)$
      $g \leftarrow \frac{g_{\mathrm{clip}}}{g_{\max}} \Delta\eta$
    end if
    $\eta \leftarrow Adam(g)$ ▷ Update parameters updates with Adam
    $k' \sim \mathrm{Uniform}(B_{\mathrm{burnin}}, (1+\gamma_B)B_{\mathrm{burnin}})$ ▷ Randomize batch burn-in
    $k_0 \leftarrow k_0 + k'$
  end while
until converged.

Table 3: Fitting Parameters for Adam.

Fitting Parameter	Value	Comment
Learning rate	0.01	Adam parameter
β_1	0.1	Adam parameter
β_2	0.001	Adam parameter
g_{clip}	100	Clipping threshold
L_{burnin}	10 s	Data burn-in
B_{burnin}	0.3 s	Minibatch burn-in
γ_L	1	L_{burnin} noise factor
γ_B	0.1	B_{burnin} noise factor

Note: Learning rate, β_1 and β_2 are as defined in Kingma and Ba (2014).

We also experimented with intializing the model at a stationary point (see appendix C), but in the cases we considered, it did not provide a notable improvement in computation time.

4.5 Estimating Parameters. To maximize the likelihood, we used Adam (Kingma & Ba, 2014), a momentum-based stochastic gradient descent algorithm, for which gradients were computed automatically with Theano (Team et al., 2016) (see section 4.9). Training parameters are listed in Table 3.

Despite the similarities, there remain important practical differences between fitting the mesoscopic model and training a recurrent neural network (RNN). Notably, RNN weights are more freely rescaled, allowing the use of single precision floating-point arithmetic. In the case of the mesoscopic model, the dynamic range is wider, and we found it necessary to use double precision.

Compared to a neural network, the mesoscopic update equations (see equations E.1 to E.23) are also more expensive to compute, in our case slowing parameter updates by at least an order of magnitude.

The sub-sequences of data ("minibatches") used to train an RNN are usually selected at random: at each iteration, a random time step k_0 is selected, from which the next B_{burnin} data points are used for burn-in and the following B data points form the minibatch. This becomes problematic when long burn-in times are required, not only because it requires long computation times but also because it wastes a lot of data. We addressed this problem by keeping the state across iterations (see algorithm 2), since this is a good guess of what it should be after updating the parameters, it reduces the required burn-in time by an order of magnitude. However, this requires batches to follow one another, breaking the usual assumption that they are independently selected. In practice, this seemed not to be a problem; in anecdotal comparisons, we found that training with either randomly selected batches and stationary initialization (see appendix C) or sequential batches and silent initialization (see algorithm 1) required comparable numbers of iterations to converge to similar parameter values. Computation time in the case of random batches, however, was much longer.

We also found that bounding the gradient helped make inference more robust. We set maximum values for each gradient component and rescaled the gradient so that no component exceeded its maximum (see algorithm 2, lines 7 to 10).

Maximizing the posterior rather than the likelihood by multiplying the latter by parameter priors (to obtain the MAP estimate rather than the MLE) helped prevent the fit from getting stuck in unphysical regions far from the true parameters, where the likelihood may not be informative. We used noninformative priors (see Table 7) so as to ensure that they did not artificially constrain the fits. Fits were also initialized by sampling from the prior.

Choosing adequate external inputs may also affect fit performance, as in general, sharp stimuli exciting transients on multiple timescales tend to be more informative than constant input (Iolov, Ditlevsen, & Longtin, 2017). That said, even under constant input, the fluctuations in a finite-sized neuron population still carry some information, and anecdotal evidence suggests that these can be sufficient to infer approximate model parameters. In this article, we used a sinusoidal input with frozen white noise to train the mesoGIF model; with only one dominant timescale, this input is more informative than constant input but far from optimal for the purpose

Table 4: Specification of the MCMC Sampler.

Algorithm	`HamiltonianMC` (PyMC3; Salvatier et al., 2016)
Step scale	0.0025
Path length	0.1
Tuning steps	20
Initialization	`jitter+adapt_diag`
Start	η_{MAP} estimate
Number of samples	2000
Total run time	201 h

of fitting. This made it a reasonable choice for computing baseline performance measures.

Finally, to allow fits to converge, it is essential to avoid fitting any ill-defined or degenerate parameters. For example, as explained in section 2.2, we fixed the parameters u_{rest} and u_{th} because the mesoGIF model is invariant under a rescaling of the voltage; for simplicity we also fixed u_{r} and R even though this was not strictly necessary. The parameters w and p are similarly degenerate (see equation E.7), and we fixed p. The parameters N, Δ, and t_{ref} are effectively discrete (either in numbers of neurons or time bins), and they were also fixed to simplify the implementation. Table 2 summarizes the inferred and noninferred parameters.

4.6 Estimating the Posterior. The posteriors in section 2.5 were obtained using Hamiltonian Monte Carlo (Betancourt & Girolami, 2013; Neal, 2012) (HMC). Having expressed the likelihood with Theano made it straightforward to use the implementation in PyMC3 (Salvatier, Wiecki, & Fonnesbeck, 2016)—HamiltonianMC—to sample the likelihood; the sampling parameters we used are listed in Table 4.

Although straightforward, this approach pushes the limit of what can be achieved with currently implemented samplers: because the likelihood of this model is expensive to evaluate, even coarse distributions can take hours to obtain. In addition, the large state vector required sufficiently large amounts of memory to make the automatically tuned NUTS (Hoffman & Gelman, 2014) sampler impractical. (NUTS stores the most recent states in order to tune the sampling parameters.) In an application with experimental data, one would want to reserve sufficient computational resources to perform at least basic validation of the obtained that posterior, using, for example, the methods described in Gelman et al. (2014) and Talts, Betancourt, Simpson, Vehtari, and Gelman (2018).

In order for samplers to find the high-probability-density region in finite time, we found it necessary to initialize them with the MAP estimate.

This also ensured that their mass matrix was tuned on an area of the posterior with appropriate curvature. In applications where the posterior has multiple modes, one should be able to identify them from the collection of fits. The high-probability-density region around each mode should then be sampled separately, integrated, and combined with the others to obtain the full posterior. (See, e.g., van Haasteren, 2014, for integration methods for MCMC chains.)

Finally, as with parameter optimization, we found that the use of at least double-precision floats was required in order to obtain consistent results.

4.7 Measuring Performance. In order to assess the performance of our inference method, we quantified the discrepancy between a simulation using ground-truth parameters and another using inferred parameters; the same input was used for both simulations and was different from the one used for training. Following Augustin et al. (2017), discrepancy was quantified using both correlation (ρ) and root mean square error (RMSE); these are reported according to the amount of data L used to train the model, which may be given in either time bins or seconds.

The correlation between activity traces from the ground-truth and inferred models, respectively, $A^{\text{true}}(t)$ and $\hat{A}^{(L)}(t)$, was obtained by computing the per trial Pearson coefficient for each of the M populations and averaging the results across populations to report a single value:

$$\rho(A^{\text{true}}, \hat{A}^{(L)}) = \frac{1}{M}\sum_{\alpha=1}^{M} \frac{\left\langle \left(A_\alpha^{\text{true}} - \langle A_\alpha^{\text{true}}\rangle\right)\left(\hat{A}_\alpha^{(L)} - \langle \hat{A}_\alpha^{(L)}\rangle_k\right)\right\rangle_k}{\sqrt{\left\langle \left(A_\alpha^{\text{true}} - \langle A_\alpha^{\text{true}}\rangle_k\right)^2\left(\hat{A}_\alpha^{(L)} - \langle \hat{A}_\alpha^{(L)}\rangle_k\right)^2\right\rangle_k}}. \quad (4.24)$$

Here, brackets indicate averages over time,

$$\langle A\rangle_k := \frac{1}{L'}\sum_{k=k_0}^{k_0+L'} A^{(k)},$$

with k a discretized time index. The initial time point k_0 sets the burn-in period; in all calculations that follow, we set it to correspond to 10 s to ensure that any artifacts due to the initialization have washed away. The value of L' need not be the same as L, and we set it to 9000 (corresponding to 9 s) for all discrepancy estimates.

As with correlation, the per trial RMSE was averaged across populations:

$$\text{RMSE}(A^{\text{true}}, \hat{A}^{(L)}) := \sqrt{\frac{1}{M}\sum_{\alpha=1}^{M}\left\langle \left(\hat{A}_\alpha^{(L)} - A_\alpha^{\text{true}}\right)^2\right\rangle_k}. \quad (4.25)$$

Because the models are stochastic, equation 4.24 and 4.25 describe random variables. Thus, for each of our results, we generated ensembles of realizations $\{A^{\text{true},r}\}_{r=1}^{R_1}$, $\{A^{\text{true},r'}\}_{r=1}^{R_2}$, and $\{\hat{A}^r\}_{r=1}^{R_3}$, each with a different set of random seeds. We compute the ρ and RMSE for the $R_1 \times R_2$ pairs $(A^{\text{true},r}, \hat{A}^r)$, as well the $R_1 \times R_3$ combinations $(A^{\text{true},r}, A^{\text{true},r'})$, from which we empirically estimate the mean and standard deviation of those measures. Values for the pairs $(A^{\text{true},r}, A^{\text{true},r'})$ provide an estimate of the best achievable value for a given measure.

Another way to address the stochasticity of these measures is to use *trial-averaged* traces:

$$\bar{\rho}(L) = \rho(\bar{A}^{\text{true}}, \hat{\bar{A}}), \tag{4.26}$$

$$\overline{\text{RMSE}}(L) = \text{RMSE}(\bar{A}^{\text{true}}, \hat{\bar{A}}); \tag{4.27}$$

where the trial-averaged activity,

$$\bar{A}_\alpha^{(k)} := \frac{1}{R} \sum_{r=1}^{R} A_\alpha(t_k | \mathcal{H}_{t_k}^r),$$

is as in equation 2.5. Because trial-averaged measures only provide a point estimate, we used bootstrapping to estimate their variability. We resampled the ensemble of realizations with replacement to generate a new ensemble of same size R and repeated this procedure 100 times. This yielded a set of R measures (either $\bar{\rho}$ or $\overline{\text{RMSE}}$), for which we computed the sample standard deviation. Note that in contrast to per trial measures, errors on trial-averaged measurements vanish in the limit of large number of trials R and thus are not indicative of the variability between traces.

We found the pair of measures ($\bar{\rho}$, RMSE) (see equations 4.25 and 4.26) to provide a good balance between information and conciseness (see section 2.3). We generally used $R_1 = R_2 = 50$ and $R_3 = 100$ for the ensembles, with the exception of Figure 4, where $R_1 = R_2 = R_3 = 20$. We also ensured that sets of trial-averaged measures use the same number of trials, to ensure comparability.

4.8 Stimulation and Integration Details. All external inputs used in this article are shared within populations and frozen across realizations. They are distinct from the escape noise (see equations 2.1 and 2.4), which is *not* frozen across realizations.

4.8.1 Sine-Modulated White Noise Input. For inferring parameters in all our work, we generated training data with a sine-modulated stimulus of the form

Table 5: Parameters for the Sine-Modulated Input.

	Two-Population Model		Four-Population Model				
	E	I	L2/3e	L2/3i	L4e	L4i	Unit
B	0.25	0.1	0.0	0.0	0.25	0.1	mA
ω	2.0	2.0	2.0	2.0	2.0	2.0	–
q	4.0	4.0	4.0	4.0	4.0	4.0	mA

Table 6: Parameters for the OU-Process Input (Equation 4.29).

	Two-Population Model		Four-Population Model				Unit
	E	I	L2/3e	L2/3i	L4e	L4i	
μ_{OU}	0.1	0.05	1	1	0	1	mA
τ_{OU}	1	1	2	2	–	2	s
q	0.125	0.125	0.5	0.5	0	0.5	mA
$I_{\mathrm{test}}(0)$	0.1	0.05	0.5	0.5	0	0.5	mA

$$I_{\mathrm{ext}}(t) = B\sin(\omega t)\cdot(1 + q\xi(t)), \tag{4.28}$$

where $\xi(t)$ is the output of a white noise process with $\langle \xi(t)\xi(t')\rangle = \delta(t-t')$. This input was chosen to be weakly informative in order to provide a baseline for the inference procedure. The values of B, ω and q are listed in Table 5. The integration time step was set to 0.2 ms for microscopic simulations and 1 ms for mesoscopic simulations. We then tested the fitted model with the inputs described below.

4.8.2 OU Process Input. Fit performance in sections 2.3 and 2.6 was measured using an input produced by an Ornstein-Uhlenbeck (OU) process defined by

$$\frac{dI_{\mathrm{test}}}{dt} = -\frac{(I_{\mathrm{test}} - \mu_{\mathrm{OU}})}{\tau_{\mathrm{OU}}}\,dt + q\sqrt{\frac{2}{\tau_{\mathrm{OU}}}}dW. \tag{4.29}$$

Here μ_{OU}, τ_{OU} and q, respectively, set the mean, correlation time, and noise amplitude of the input, while dW denotes increments of a Wiener process. The parameter values and initial condition ($I_{\mathrm{test}}(0)$) are listed in Table 6.

4.8.3 Impulse Input. We further tested the generalizability of the four-population model using an input composed of sharp, synchronous ramps. As the transient response is qualitatively different from the sinusoidal oscillations used to fit the model, this is a way of testing the robustness of the

inferred parameters to extrapolation. The input had the following form:

$$I_\alpha(t) = \sum_{t_0 \in \mathcal{T}} \mathcal{J}_{t_0}(t) \tag{4.30}$$

$$\mathcal{J}_{t_0}(t) = \begin{cases} B\left(1 - \frac{|t-t_0|}{d}\right) & \text{if } |t - t_0| \leq d, \\ 0 & \text{otherwise.} \end{cases} \tag{4.31}$$

The input was generated with $d = 0.15$ s, $B = (0, 0, 0.6, -0.6)$ mA. Impulses were placed at

$$\mathcal{T} = \{11.0, 11.7, 12.2, 12.9, 14.1, 14.5, 15.5, 15.8, 16.2, 16.8\} \text{ s}.$$

4.8.4 Numerical Integration. For all simulations of the mesoGIF model, we used a time step of 1 ms. We also used a 1 ms time step when inferring parameters. Simulations of the microscopic GIF model require finer temporal resolution, and for those we used time steps of 0.2 ms. In order to have the same inputs at both temporal resolutions, they were generated using the finer time step and coarse-grained by averaging.

We used the Euler-Maruyama scheme to integrate inputs; the GIF and mesoGIF models are given as update equations of the form $A(t + \Delta t) = F(A(t))$, and thus already define an integration scheme.

4.9 Software. We developed software for expressing likelihoods of dynamical systems by building on general-purpose machine learning libraries: `Theano_shim` (`https://github.com/mackelab/theano_shim`) is a thin layer over the numerical back-end, allowing one to execute the same code either using Theano (Team et al., 2016) or Numpy (Jones et al., 2001). `Sinn` (`https://github.com/mackelab/sinn`) makes use of `theano_shim` to provide a back-end-agnostic set of high-level abstractions to build dynamical models. Finally, a separate repository (`https://github.com/mackelab/fsGIF`) provides the code specific to this article.

Appendix A: Priors and Parameter Values

For both microscopic and mesoscopic models, unless otherwise specified in the text, we used the same parameter values as our ground-truth values. Values are listed in Table 7 and are based on those given in Schwalger et al. (2017), and we follow the recommendation there of adjusting resting potentials u_{rest} to maintain realistic firing rates. To facilitate simulations, we also reduced the population sizes by a factor of 50 and correspondingly upscaled the connectivity weights by a factor of $\sqrt{50}$, to maintain a balanced E-I network (Vogels, Rajan, & Abbott, 2005).

Table 7: Default Parameter Values and Priors.

Parameter	Value: Two-Population Model	Value: Four-Population Model	Unit	Prior Distribution
Population labels	E, I	L2/3e, L2/3i, L4e, L4i		
N	(438, 109)	(413, 116, 438, 109)		
R	(19, 11.964)	(0, 0, 19, 11.964)	Ω	
u_{rest}	(20, 19.5)	(18, 18, 25, 20)	mV	
p	$\begin{pmatrix} 0.0497 & 0.1350 \\ 0.0794 & 0.1597 \end{pmatrix}$	$\begin{pmatrix} 0.1009 & 0.1689 & 0.0437 & 0.0818 \\ 0.1346 & 0.1371 & 0.0316 & 0.0515 \\ 0.0077 & 0.0059 & 0.0497 & 0.1350 \\ 0.0691 & 0.0029 & 0.0794 & 0.1597 \end{pmatrix}$		
w	$\begin{pmatrix} 2.482 & -4.964 \\ 1.245 & -4.964 \end{pmatrix}$	$\begin{pmatrix} 1.245 & -4.964 & 1.245 & -4.964 \\ 1.245 & -4.964 & 1.245 & -4.964 \\ 1.245 & -4.964 & 2.482 & -4.964 \\ 1.245 & -4.964 & 1.245 & -4.964 \end{pmatrix}$	mV	$w \sim \mathcal{N}\left(0, 4^2\right)$
τ_m	(0.01, 0.01)	(0.01, 0.01, 0.01, 0.01)	s	$\log_{10} \tau_m \sim \mathcal{N}(-2, 2^2)$
t_{ref}	(0.002, 0.002)	(0.002, 0.002, 0.002, 0.002)	s	
u_{th}	(15, 15)	(15, 15, 15, 15)	mV	$u_{\text{th}} \sim \mathcal{N}(15, 10^2)$
u_{r}	(0, 0)	(0, 0, 0, 0)	mV	$u_{\text{r}} \sim \mathcal{N}(0, 10^2)$
c	(10, 10)	(10, 10, 10, 10)	Hz	$c \sim \Gamma(2, 5)$
Δ_{u}	(5, 5)	(5, 5, 5, 5)	mV	$\Delta_{\text{u}} \sim \Gamma(3, 1.5)$
Δ	0.001	0.001	s	
τ_s	(0.003, 0.006)	(0.003, 0.006, 0.003, 0.006)	s	$\log_{10} \tau_s \sim \mathcal{N}(-3, 3^2)$
J_θ	(1.0, 0)	(1.0, 0, 1.0, 0)	mV	$J_\theta \sim \Gamma(2, 0.5)$
τ_θ	(1.0, –)	(1.0, –, 1.0, –)	s	$\log_{10} \tau_\theta \sim \mathcal{N}(-1, 5^2)$

Notes: Symbols are the same as in Table 2. Most values are those given in Tables 1 and 2 of Schwalger et al. (2017). Priors are given as scalar distributions because they are the same for all components. The pdf of $\Gamma(\alpha, \theta)$ is $\frac{x^{\alpha-1} e^{-x/\theta}}{\theta^\alpha \Gamma(\alpha)}$.

Table 8: Distribution Parameters for the Heterogeneous Model.

	Distribution Parameter	
Heterogeneous Model Parameter	μ	σ
$\log_{10} \tau_{m,E}$	-1.6	0.5
$\log_{10} \tau_{m,I}$	-1.8	0.5
$\log_{10} \tau_{\theta,E}$	-0.7	0.5

Notes: Each parameter was sampled from a log-normal distribution $\log_{10} \mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2. No adaptation was modeled in the inhibitory population, so $\tau_{\theta,I}$ was not sampled.

Table 9: Inferred Parameters for a Heterogeneous Population.

Parameter	Inferred Value	Average Heterogeneous Value	Unit
w	$\begin{pmatrix} 1.59 & -5.05 \\ 0.73 & -3.43 \end{pmatrix}$	$\begin{pmatrix} 2.482 & -4.964 \\ 1.245 & -4.964 \end{pmatrix}$	mV
τ_m	(0.011, 0.008)	(0.056, 0.046)	s
c	(5.05, 5.22)	(10, 10)	Hz
Δ_u	(5.09, 4.09)	(5, 5)	mV
τ_s	(0.0046, 0.0109)	(0.003, 0.006)	s
J_θ	(0.538, 0)	(1.0, 0)	mV
τ_θ	(0.131, –)	(0.380, –)	s

Notes: Values are given in vector format, as (η_E, η_I). Corresponding average values for the heterogeneous microscopic model are given for comparison. (The heterogeneous model was homogeneous in all parameters except τ_m and τ_θ.)

Prior distributions on inferred parameters were set sufficiently broad to be considered noninformative. Prior distributions are independent of the population so as to ensure that any inferred feature (e.g., excitatory versus inhibitory connections) is due to the data.

The two-population heteregeneous model was obtained by sampling similar but tighter distributions as the prior (see Table 8). Only membrane and adaptation time constants were sampled; other parameters were as in Table 7.

Appendix B: Inferred Parameters

The inferred parameters for the heteregeneous two-population model (see section 2.4) show an overall reduction of connection strengths and time constants, compared with ground-truth values (see Table 9). For the

Table 10: Inferred Values for the Four-Population Model.

	MAP				Theory			
	L2/3e	*L2/3i*	*L4e*	*L4i*	*L2/3e*	*L2/3i*	*L4e*	*L4i*
$w_{L2/3e\leftarrow\cdot}$	0.734	−5.629	1.546	−5.292	1.245	−4.964	1.245	−4.964
$w_{L2/3i\leftarrow\cdot}$	1.181	−5.406	1.419	−4.294	1.245	−4.964	1.245	−4.964
$w_{L4e\leftarrow\cdot}$	1.528	−0.637	2.058	−4.213	1.245	−4.964	2.482	−4.964
$w_{L4i\leftarrow\cdot}$	0.174	1.112	1.046	−3.994	1.245	−4.964	1.245	−4.964
τ_m	0.016	0.015	0.008	0.009	0.010	0.010	0.010	0.010
c	16.717	18.170	9.020	9.680	10.000	10.000	10.000	10.000
Δu	7.435	6.453	4.750	4.420	5.000	5.000	5.000	5.000
τ_s	0.001	0.006	0.002	0.009	0.003	0.006	0.003	0.006
J_θ	0.232	—	0.967	—	1.000	—	1.000	—
τ_θ	0.425	—	1.596	—	1.000	—	1.000	—

Notes: The values for the homogeneous microscopic model used in Figures 7 and 8 are listed on the right. Theory predicts these to be the best parameterizations for the mesoscopic model and thus should be recovered by maximizing the posterior (MAP values). Since L2/3 receives no external input in the training data, the inferred parameters for those populations are understandably further from theory.

homogeneous four-population model (see section 2.6), the tendency is less systematic (see Table 10).

Appendix C: Alternative Initialization Scheme

Compared to the silent initialization (see section 4.4), the stationary initialization described by algorithm 3 finds a more realistic initial state, which reduces the burn-in time required by about an order of magnitude. This makes it more practical when minibatches are selected at random, and we used this scheme to validate algorithm 2 (see section 4.5). However, in general, we found the computational gain to be offset by the added cost of solving a self-consistent equation for each batch.

Algorithm 3: Stationary initialization scheme.

1: $A^*_\alpha \leftarrow$ Solve Equation (69)
2: $n_\alpha \leftarrow A^*_\alpha N_\alpha \Delta t$
3: $h_\alpha, y_{\alpha\beta}, u_{\alpha,i}, g_\alpha, \lambda_{\text{free},\alpha}, \lambda_{\alpha,i}, x_\alpha, z_\alpha, m_{\alpha,i}, v_{\alpha,i} \leftarrow$ Evaluate Eqs. (47–63) with A^*_α

C.1 Stationary Initialization. Assuming zero external input, we find a self-consistent equation for the stationary activity A^* (see appendix I). After solving numerically for A^*, the other state variables are then easily computed.

Table 11: Definition of Parameter Subsets for the Two-Population Model.

Subset Label	Included Parameters
η_1	$\{w_{EE}, w_{EI}, w_{IE}, w_{II}\}$
η_2	$\{\tau_{\theta,E}, J_{\theta,E}\}$
η_3	$\eta_1 \cup \eta_3 \cup \{c_E, c_I, \Delta_{\mathrm{u},E}, \Delta_{\mathrm{u},I}, \tau_{m,E}, \tau_{m,I}, \tau_{\mathrm{s},E}, \tau_{\mathrm{s},I}\}$

Note: There are only two adaptation parameters because inhibitory populations have no adaptation in this model.

Appendix D: Data Requirements for Different Parameter Sets

In section 2.3, we showed that less than 10 s of data were sufficient to infer the parameters of the two-population mesoGIF model. Of course, the exact data requirements will depend on how many parameters we need to infer and which they are (e.g., w versus τ_m).

To explore this issue, we repeated the inference procedure for the parameter subsets listed in Table 11, performing 24 fits for each subset using different amounts of data. Subsets η_1 and η_2 parameterize, respectively, the connectivity and the adaptation, while η_3 is the full set used for Figure 4. A similar figure to Figure 4 with all three subsets is shown in Figure 9A.

With the smaller subsets (η_1, η_2), 1.25 s of data was sufficient to get good accuracy of the inferred dynamics (see Figure 9A). However working with such small amounts of data incurs a substantial computational cost. First, the fits converge less consistently, thus requiring more fits to find a good estimate of the MAP (see Figures 9B–D, left). Second, the algorithm optimizations making use of the longer traces (see section 4.5) are no longer as effective, making each iteration slower on average.

Since we know the ground-truth parameters, we can further estimate the expected error by computing the relative difference between true and inferred parameter values. For a parameter η and its estimate $\eta^{(L)}$ obtained by using L seconds of data, this is calculated as

$$\Delta_{\mathrm{rel}}\left(\hat{\eta}^{(L)}\right) := \left|\frac{\hat{\eta}^{(L)} - \eta}{\eta}\right|. \tag{D.1}$$

The number of fits required to achieve this performance will vary according to the nature and number of parameters; indeed, with more parameters to infer, we found that fits terminated further from the true values. A simple way to quantify the uncertainty of any one particular fit is the sample standard deviation σ_η of the set of found optima from a collection of fits. In order to make the σ_η comparable between parameters, we normalized by the parameter mean μ_η to obtain the coefficient of variation:

$$\left|CV(\eta^{(L)})\right| \stackrel{\mathrm{def}}{=} \left|\sigma_{\eta^{(L)}}/\mu_{\eta^{(L)}}\right|. \tag{D.2}$$

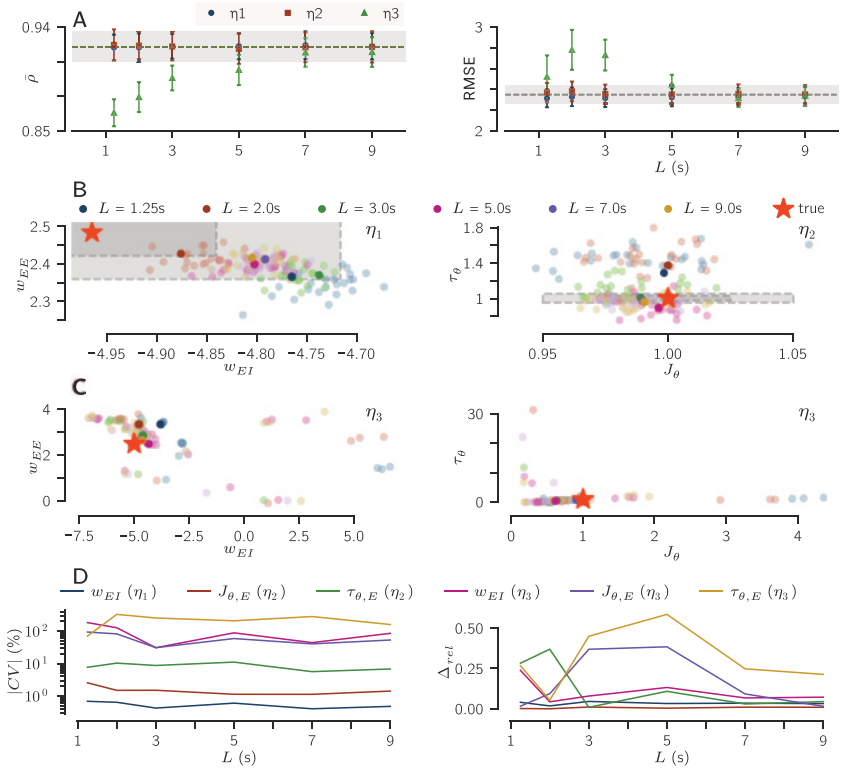

Figure 9: Fits of many parameters are less consistent. (A) As the number of inferred parameters is increased, more data are required to estimate them. η_1, η_2 are parameter sets corresponding, respectively, to connectivity and adaptation. $\eta_3 \supset (\eta_1 \cup \eta_2)$ is the set of all parameters. Definitions in Table 11. (B) Results from 24 fits for subsets η_1 (left) and η_2 (right) for different amounts L of data. The star indicates the true parameters and gray boxes the 5% and 10% relative errors Δ_{rel}. Fits cluster around the MAP, which for finite amounts of data will not exactly coincide with the ground-truth values. Darker dots indicate the fit with the highest likelihood. The consistency of estimates for the adaptation parameters, with $\tau_{\theta,E} = 1$ s, is particularly improved with longer data traces. (C) Same as in panel B but all parameters were simultaneously inferred. The reduced consistency is noticeable by the change of scale, at which the 5% and 10% relative error boxes are not visible. (D) When going from inferring smaller (η_1, η_2) to larger (η_3) subsets of parameters, the increase in relative error for the same number of fits is relatively modest (right) compared to the wider area in parameter space to which fits converge (left). Figure traces statistics for different parameters as a function of the amount of data (L) and the subset of parameters that were fit simultaneously (η_1–η_3). Values for all data and subset combinations are given in Tables 12 and 13.

Table 12: Relative Error for the Fits Shown in Section 2.3.

		L					
Subset	Parameter	1.25	2.00	3.00	5.00	7.00	9.00
η_1	w_{EE}	0.047	0.023	0.045	0.034	0.029	0.027
	w_{EI}	0.040	0.018	0.046	0.033	0.035	0.032
	w_{IE}	0.013	0.038	0.000	0.001	0.005	0.024
	w_{II}	0.018	0.005	0.022	0.018	0.012	0.005
η_2	$J_{\theta,E}$	0.002	0.000	0.011	0.004	0.010	0.009
	$\tau_{\theta,E}$	0.283	0.370	0.009	0.108	0.030	0.045
η_3	w_{EE}	0.345	0.348	0.151	0.001	0.084	0.067
	w_{EI}	0.238	0.043	0.079	0.132	0.067	0.072
	w_{IE}	0.017	0.556	0.630	0.427	0.244	0.178
	w_{II}	0.070	0.495	0.503	0.326	0.180	0.136
	$J_{\theta,E}$	0.016	0.094	0.369	0.385	0.092	0.016
	$\tau_{\theta,E}$	0.267	0.054	0.450	0.586	0.248	0.213
	c_E	0.420	0.376	0.469	0.382	0.239	0.160
	c_I	2.825	0.006	0.133	0.142	0.128	0.161
	Δu_E	0.215	0.182	0.090	0.086	0.092	0.052
	Δu_I	0.520	0.338	0.466	0.302	0.129	0.058
	$\tau_{m,E}$	0.190	0.117	0.057	0.177	0.052	0.037
	$\tau_{m,I}$	2.590	0.619	0.430	0.393	0.235	0.219
	$\tau_{s,E}$	0.744	0.101	0.038	0.101	0.039	0.142
	$\tau_{s,I}$	0.271	0.119	0.138	0.132	0.081	0.081

Relative error and CV values for all parameter subsets are listed in Tables 12 and 13.

Appendix E: Mesoscopic Update Equations

This section first describes the quantities composing the state vector for the mesoGIF model and then lists the equations used for this article. All equations are for discretized time, and we use a superscript (k) to indicate the kth time step. For derivations and a fuller discussion of the variables involved, see Schwalger et al. (2017).

E.1 Construction of the State Vector. In order to obtain a finite state vector (see section 4.2), neurons are divided into two categories: free and refractory; the assignment of neurons to either category changes over time, following a discretized form of the transport equation, 4.10.

Refractory neurons are still in the absolute or relative refractory period caused by their last spike, and thus have a higher firing threshold. Since the height of the threshold is dependent on that spike's time, we track a vector $m_\alpha^{(k)}$, indexed by the age l. We define the scalar $m_{\alpha,l}^{(k)}$ as our estimate of the number of neurons at time t_k that last fired at time t_{k-l}. A vector $v_\alpha^{(k)}$

Table 13: Coefficients of Variation for the Collections of Fits Shown in Section 2.3.

		L					
Subset	Parameter	1.25	2.00	3.00	5.00	7.00	9.00
η_1	w_{EE}	1.33	1.05	0.95	0.72	0.61	0.82
	w_{EI}	0.69	0.63	0.43	0.62	0.40	0.48
	w_{IE}	1.38	1.48	1.53	1.75	1.90	1.36
	w_{II}	0.40	0.46	0.56	0.39	0.61	0.65
η_2	$J_{\theta,E}$	2.56	1.49	1.50	1.13	1.13	1.42
	$\tau_{\theta,E}$	7.70	10.33	8.75	11.11	5.64	6.82
η_3	w_{EE}	31.63	31.73	29.26	37.84	38.29	31.49
	w_{EI}	183.77	127.48	30.84	88.90	44.35	85.80
	w_{IE}	64.08	100.98	1833.28	1143.47	1489.30	2405.89
	w_{II}	53.90	65.92	36.25	78.14	37.70	55.10
	$J_{\theta,E}$	94.03	82.51	30.99	59.25	40.58	53.40
	$\tau_{\theta,E}$	70.53	329.84	255.21	209.10	282.68	160.36
	c_E	83.08	65.81	37.16	39.42	48.50	47.80
	c_I	29.31	28.84	34.10	52.49	49.66	52.50
	Δu_E	14.98	19.16	6.11	11.37	7.58	8.84
	Δu_I	32.84	30.33	13.28	41.22	21.86	34.30
	$\tau_{m,E}$	429.43	420.71	428.10	431.68	441.05	444.64
	$\tau_{m,I}$	256.36	269.36	346.42	337.09	314.64	288.72
	$\tau_{s,E}$	427.66	441.61	447.11	446.85	446.84	447.20
	$\tau_{s,I}$	238.26	240.74	65.92	262.20	71.13	295.76

similarly tracks the variance of that estimate. The adaptation of the neurons depends on their age, such that their firing rate is also given by a vector, $\lambda_\alpha^{(k)}$. With an adaptation timescale τ_θ of 1 s and time steps of 1 ms, these vectors each comprise around $K = 1000$ age bins. For a more detailed discussion on properly choosing K, see equation 86 in Schwalger et al. (2017).

Free neurons, meanwhile, have essentially forgotten their last spike: their firing threshold has relaxed back to its resting state, and so they can be treated as identical, independent of when that last spike was. One scalar per population, $\lambda_{\text{free},\alpha}^{(k)}$, suffices to describe their firing rate. Scalars $x_\alpha^{(k)}$ and $z_\alpha^{(k)}$ respectively track the estimated mean and variance of the number of free neurons.

In the case of an infinite number of neurons, the firing rates $\lambda_\alpha^{(k)}$ and $\lambda_{\text{free},\alpha}^{(k)}$ would be exact, but for finite populations, a further correction $P_{\Lambda,\alpha}^{(k)}$ must be made to account for statistical fluctuations. Combining $\lambda_{\text{free},\alpha}^{(k)}$, $\lambda_\alpha^{(k)}$, and $P_{\Lambda,\alpha}^{(k)}$, one can compute $\bar{n}_\alpha^{(k)}$, the expected number of spikes at t_k. The definition of $n^{(k)}$ then follows as described in section 4.2.

For both refractory and free neurons, the dependency of their time evolution on the spiking history of the network is taken into account by

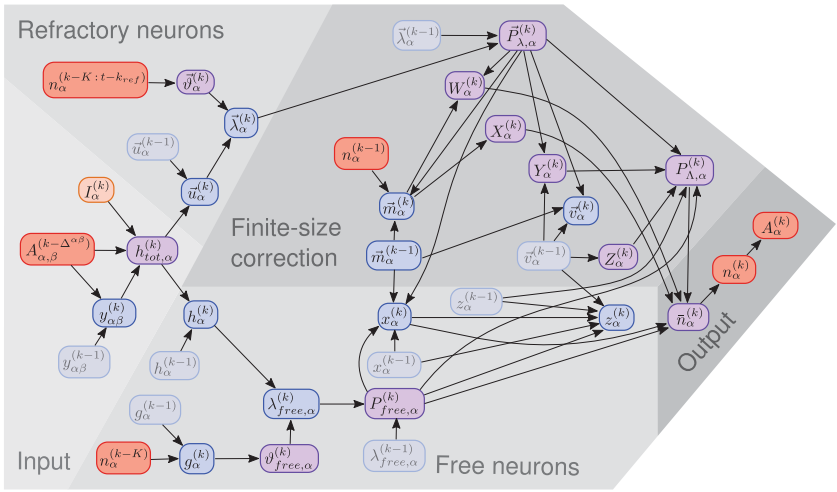

Figure 10: Graphical representation of the mesoscopic model. An arrow $x \to y$ indicates that x is required in order to compute y. Red (orange) boxes indicate observed variables (input) with the external input shown in lighter orange. Variables in blue boxes must be stored until the next iteration, and along with the activity A, they form the model's state. Intermediate variables shown in purple do not need to be stored. Indices in parentheses indicate the time step, Greek letters the population index. During simulation, mesoscopic model parameters (not shown, but determine the computations along arrows) are fixed, and mesoscopic output variables $A_\alpha^{(k)}$ are generated in a given time step; these values form the input for the next time step. During inference, the input is obtained from the training data and is used to compute the sequence of binomial means $\bar{n}_\alpha^{(k)}$. These outputs, along with the observed outputs in the training data, are used to compute the likelihood. The gradient descent algorithm then changes the model parameters after each batch of training data to maximize the likelihood. See also Schwalger et al. (2017, Figure 12).

convolving the population activities (one per population) with synaptic, membrane, and adaptation kernels. Following Schwalger et al. (2017), we express these as exponential filters; this allows the associated convolutions to be respectively replaced by three additional dynamic variables y, h, and g, making forward simulations more efficient. Replacing temporal filters by dynamic variables has the additional important benefit of making the dynamics Markovian when we consider them as updates on a state $S^{(k)}$, composed of the concatenation of the blue variables in Figure 10:

$$S^{(k)} := \left(n^{(k)}, y^{(k)}, g^{(k)}, h^{(k)}, \boldsymbol{u}^{(k)}, \boldsymbol{\lambda}^{(k)}, \lambda_{\text{free}}^{(k)}, \boldsymbol{m}^{(k)}, \boldsymbol{v}^{(k)}, x^{(k)}, z^{(k)}\right). \tag{E.1}$$

For clarity, we have here typeset in bold the components of $S^{(k)}$ with both population and age dimensions.

E.2 Kernels and Link Function. The equations below follow from Schwalger et al. (2017) after setting the synaptic filter to an exponential: $\epsilon_{\alpha\beta}(s) = \Theta(s - \Delta_{\alpha\beta})e^{-(s-\Delta)/\tau_{s,\beta}}/\tau_s(\beta)$. They depend on the inverse link function f, relating membrane potential to spiking probability and a refractory/adaptation kernel θ. Throughout this work we used

$$f_\alpha(u') = c_\alpha \exp(u'/\Delta_{u,\alpha}) \tag{E.2}$$

and

$$\theta_\alpha(t) = \begin{cases} \infty & \text{if } t < t_{\text{ref},\alpha}, \\ \frac{J_{\theta,\alpha}}{\tau_{\theta,\alpha}} e^{(t-t_{\text{ref},\alpha})/\tau_{\theta,\alpha}} & \text{otherwise.} \end{cases} \tag{E.3}$$

The quasi-renewal kernel (Naud & Gerstner, 2012) used below is defined as

$$\tilde{\theta}_\alpha(t) = \Delta_{u,\alpha}\left[1 - e^{-\theta_\alpha(t)/\Delta_{u,\alpha}}\right]. \tag{E.4}$$

State vectors assign the index 0 to the time Δt, such that they run from $\theta_0 = \theta(\Delta t)$ to $\theta_K = \theta((K+1)\Delta t)$, with $K \in \mathbb{N}$. We define k_{ref} to be the lengths of the absolute refractory periods in time bins, that is, $t_{\text{ref},\alpha} = k_{\text{ref},\alpha}\Delta t$, for each population α.

E.3 Update Equations. Total input

$$h^{(k+1)} = u_{rest} + (u^{(k)} - u_{rest})e^{-\Delta t/\tau_m} + h_{tot}, \tag{E.5}$$

$$y_{\alpha\beta}^{(k+1)} = A_\beta(t_k - \Delta_{\alpha\beta}) + \left[y_{\alpha\beta}^{(k)} - A_\beta(t_k - \Delta_{\alpha\beta})\right] e^{-\Delta t/\tau_{s,\beta}}. \tag{E.6}$$

$$h_{tot,\alpha} = RI_{ext}^{(k)}\left(1 - e^{-\Delta t/\tau_{m,\alpha}}\right) + \tau_{m,\alpha}\sum_{\beta=1}^{M} p_{\alpha\beta}N_\beta w_{\alpha\beta}\Bigg\{A_\beta(t-\Delta_{\alpha\beta}) + \frac{\tau_{s,\beta}e^{-\frac{\Delta t}{\tau_{s,\beta}}}\left[y_{\alpha\beta}^{(k)} - A_\beta(t_k-\Delta_{\alpha\beta})\right] - e^{-\frac{\Delta t}{\tau_{m,\alpha}}}\left[\tau_{s,\beta}y_{\alpha\beta}^{(k)} - \tau_{m,\alpha}A_\beta(t_k - \Delta_{\alpha\beta})\right]}{\tau_{s,\beta} - \tau_{m,\alpha}}\Bigg\}. \tag{E.7}$$

Membrane potential, refractory neurons

$$u_{\alpha,i}^{(k+1)} = \begin{cases} u_{r,\alpha} & 0 \le i < k_{\mathrm{ref},\alpha}, \\ u_{rest,\alpha} + (u_{\alpha,i-1}^{(k)} - u_{rest,\alpha})e^{-\Delta t/\tau_{m,\alpha}} + h_{tot,\alpha}^{(k)} & i \ge k_{\mathrm{ref},\alpha}. \end{cases} \tag{E.8}$$

Firing threshold

$$\vartheta_{\alpha i}^{(k+1)} = \vartheta_{\mathrm{free},\alpha}^{(k+1)} + \theta_{\alpha i} + \frac{1}{N}\sum_{j=i+1}^{K} \tilde{\theta}_{\alpha,j}\Delta n_{\alpha}^{(k-j-1)}, \tag{E.9}$$

$$\vartheta_{\mathrm{free},\alpha}^{(k+1)} = u_{th,\alpha} + J_{\theta,\alpha}e^{-T/\tau_{\theta,\alpha}}g^{(k+1)}, \tag{E.10}$$

$$g_{\alpha}^{(k+1)} = e^{-\Delta t/\tau_{\theta,\alpha}}g_{\alpha}^{(k)} + (1 - e^{-\Delta t/\tau_{\theta,\alpha}})A_{\alpha}^{(k+1-K)}. \tag{E.11}$$

Firing probabilities

$$\lambda_{\mathrm{free},\alpha}^{(k)} = f(h_{\alpha}^{(k)} - \vartheta_{\mathrm{free},\alpha}^{(k)}), \quad \lambda_{\alpha i}^{(k)} = \begin{cases} 0 & 0 \le i < k_{\mathrm{ref},\alpha}, \\ f(u_{\alpha i}^{(k)} - \vartheta_{\alpha i}^{(k)}) & k_{\mathrm{ref},\alpha} \le i < K. \end{cases} \tag{E.12}$$

$$P_{\mathrm{free},\alpha}^{(k)} = 1 - e^{-\bar{\lambda}_{\mathrm{free},\alpha}^{(k)}\Delta t}, \quad P_{\lambda,\alpha i}^{(k)} = 1 - e^{-\bar{\lambda}_{\alpha i}^{(k)}\Delta t}, \tag{E.13}$$

where

$$\bar{\lambda}_{\mathrm{free},\alpha}^{(k)} = [\lambda_{\mathrm{free},\alpha}^{(k-1)} + \lambda_{\mathrm{free},\alpha}^{(k)}]/2, \tag{E.14}$$

$$\bar{\lambda}_{\alpha i}^{(k)} = [\lambda_{\alpha,i-1}^{(k-1)} + \lambda_{\alpha i}^{(k)}]/2. \tag{E.15}$$

Survival counts

$$\bar{n}_{\alpha}^{(k)} = \sum_{i=0}^{K-1} P_{\lambda,\alpha i}^{(k)}\bar{m}_{\alpha i}^{(k)} + P_{\mathrm{free},\alpha}^{(k)}x_{\alpha}^{(k)} + P_{\Lambda,\alpha}^{(k)}\left(N_{\alpha} - \sum_{i=0}^{K-1}\bar{m}_{\alpha i}^{(k)} - x_{\alpha}^{(k)}\right), \tag{E.16}$$

$$a_{\alpha}^{(k)} = \frac{\bar{n}_{\alpha}^{(k)}}{N_{\alpha}\Delta t}, \tag{E.17}$$

where

$$P_{\Lambda,\alpha}^{(k)} = \frac{\sum_{i=0}^{K-1} P_{\lambda,\alpha i}^{(k)}v_{\alpha i}^{(k)} + P_{\mathrm{free}}^{(k)}z_{\alpha}^{(k)}}{\sum_{i=0}^{K-1} v_{\alpha i}^{(k)} + z_{\alpha}^{(k)}}, \tag{E.18}$$

$$x_{\alpha}^{(k)} = \sum_{i=K}^{\infty}\bar{m}_{\alpha i}^{(k)} = (1 - P_{\mathrm{free},\alpha}^{(k)})x_{\alpha}^{(k-1)} + m_{\alpha K}^{(k)}, \tag{E.19}$$

Table 14: Performance of Four Population Models: Per Trial RMSE, Equation 4.25.

Input	Model	RMSE			
		L2/3e	L2/3i	L4e	L4i
Sine	True – micro	1.39 ± 0.03	3.46 ± 0.10	3.47 ± 0.13	4.59 ± 0.13
	Theory – meso	1.39 ± 0.04	3.37 ± 0.10	3.76 ± 0.15	4.51 ± 0.13
	MAP – meso	1.39 ± 0.03	3.49 ± 0.09	3.46 ± 0.13	4.53 ± 0.13
OU	True – micro	1.22 ± 0.03	3.14 ± 0.08	2.26 ± 0.07	5.13 ± 0.15
	Theory – meso	1.21 ± 0.03	3.06 ± 0.09	2.26 ± 0.07	4.95 ± 0.15
	MAP – meso	1.22 ± 0.03	3.11 ± 0.08	2.25 ± 0.06	5.30 ± 0.14
Impulse	True – micro	1.54 ± 0.05	3.64 ± 0.11	5.32 ± 0.46	5.11 ± 0.23
	Theory – meso	1.59 ± 0.06	3.63 ± 0.11	7.99 ± 0.66	5.88 ± 0.36
	MAP – meso	1.70 ± 0.07	3.96 ± 0.13	7.74 ± 0.59	6.00 ± 0.36

Note: Measures computed from 60 realizations of each model.

$$z_\alpha^{(k)} = \sum_{i=K}^{\infty} v_{\alpha i}^{(k)} = (1 - P_{\text{free},\alpha}^{(k)})^2 z_\alpha^{(k-1)} + P_{\text{free},\alpha}^{(k)} x_\alpha^{(k-1)} + v_{\alpha K}^{(k)}, \tag{E.20}$$

$$\bar{m}_{\alpha i}^{(k)} = \begin{cases} n_\alpha^{(k-1)} & \text{if } i = 0, \\ [1 - P_{\lambda,\alpha i}^{(k)}] \bar{m}_{\alpha,i-1}^{(k-1)} & \text{otherwise;} \end{cases} \tag{E.21}$$

$$v_{\alpha i}^{(k)} = \begin{cases} 0 & \text{if } i = 0, \\ [1 - P_{\lambda,\alpha i}^{(k)}]^2 v_{\alpha,i-1}^{(k-1)} + P_{\lambda,\alpha i}^{(k)} \bar{m}_{\alpha i-1}^{(k-1)} & \text{otherwise.} \end{cases} \tag{E.22}$$

Spike generation

$$n_\alpha^{(k)} \sim \text{Binom}(\bar{n}_\alpha^{(k)}/N_\alpha; N_\alpha). \tag{E.23}$$

This last equation is the one identified as equation 4.18 in the main text.

Appendix F: Performance of Four Population Models

Per population performance measures for the four-population model (see section 2.6) are compiled in Tables 14 and 15.

Appendix G: Posterior for the Two-Population Mesoscopic Model

In section 2.5, we estimated a 14-dimensional posterior over parameters for the two-population mesoGIF model and showed a subset of its 2D marginals. The complete set of these marginals is shown in Figure 11.

Table 15: Performance of Four Population Models: Trial-Averaged Correlation, Equation 4.26.

		$\bar{\rho}$			
Input	Model	L2/3e	L2/3i	L4e	L4i
Sine	True – micro	0.418	0.386	0.994	0.948
	Theory – meso	0.354	0.348	0.991	0.945
	MAP – meso	0.352	0.455	0.994	0.951
OU	True – micro	0.829	0.694	0.977	0.905
	Theory – meso	0.815	0.717	0.978	0.914
	MAP – meso	0.855	0.756	0.977	0.916
Impulse	True – micro	0.914	0.879	0.996	0.927
	Theory – meso	0.880	0.858	0.979	0.870
	MAP – meso	0.912	0.896	0.988	0.887

Note: Measures computed from 60 realizations of each model.

Appendix H: Fit Dynamics

When fitting to data produced with a homogeneous microscopic model, inferred parameters are consistent with those predicted by the mesoscopic theory (see Figures 12 and 13).

Appendix I: Self-Consistent Equation for the Mesoscopic Stationary State

We derive the stationary state for the case where $I_{ext} \equiv 0$. For analytical tractability, we assume that finite-size fluctuations are negligible (effectively, that N_α is very large), such that in the stationary state, the activity is constant. We denote this activity A^*.

Having no fluctuations means that expected and observed spikes are interchangeable and equal to a constant:

$$n_\alpha^{(k)} = \bar{n}_\alpha^{(k)} = A^* N_\alpha \Delta t. \tag{I.1}$$

This means that the number of spikes never overshoots or undershoots $\bar{n}_\alpha$, and the correction factor P_Λ is zero. Equivalently, we can state that

$$N = \sum_{i=0}^{K-1} n\bar{m}_i + x. \tag{I.2}$$

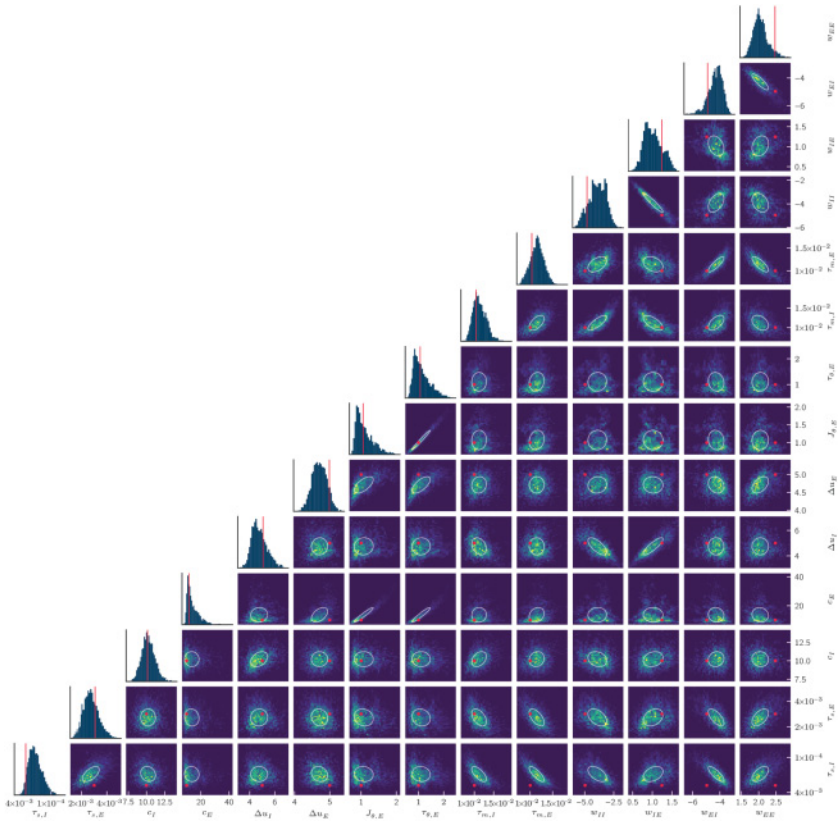

Figure 11: Full posterior for the two-population mesoscopic model. Red points indicate the true values, and ellipses trace the two-standard-deviation isoline assuming a gaussian model. Many parameter pairs show noticeable correlation, such as $\tau_{\theta,E}$ and $J_{\theta,E}$, or w_{IE} and $\Delta_{\mathrm{u},I}$.

Substituting the stationary values $A^*, h^*, \ldots$ into the equations of appendix H, we obtain equations for the stationary state—for instance,

$$h^* = u_{rest,\alpha} + \tau_m \sum_{\beta=1}^{M} p_{\alpha\beta} N_\beta w_{\alpha\beta} A^*_{\alpha\beta}, \tag{I.3}$$

$$y^*_{\alpha\beta} = A^*_{\alpha\beta}, \tag{I.4}$$

$$\vdots$$

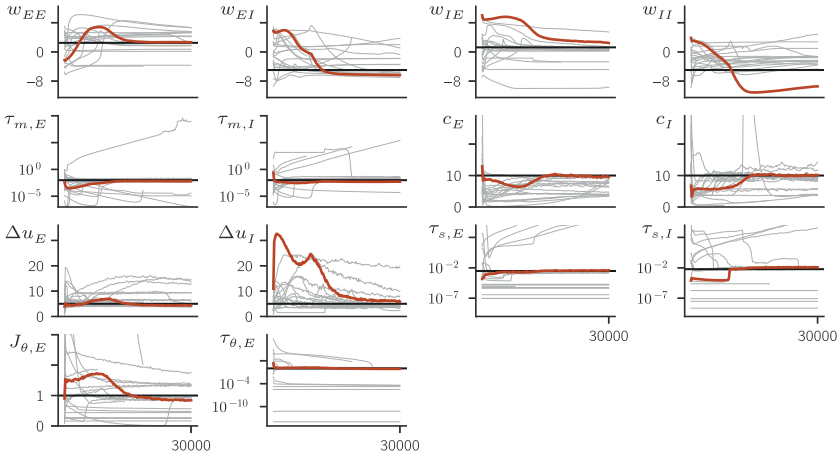

Figure 12: Fit dynamics for the two-population model. Shown are the 25 fits used to infer parameters for the two-population model in section 2.2. The fit in red is the one that found the local optimum with the highest likelihood. Fourteen parameters were inferred; black lines indicate theoretical values.

and so on. Combining these with equation I.2, we obtain a self-consistency relation,

$$1 = A^*_\alpha \Delta t \left\{ k_{\mathrm{ref},\alpha} + 1 + \sum_{i=k_{\mathrm{ref},\alpha}}^{K-1} \exp\left[- \sum_{j=k_{\mathrm{ref},\alpha}+1}^{i-1} f(a_{\alpha j} + b^\beta_{\alpha j} A^*_\beta - c_{\alpha j} A^*_\alpha)\Delta t \right] \right.$$
$$\left. + \frac{\exp\left[- \sum_{j=k_{\mathrm{ref},\alpha}}^{K-1} f(a_{\alpha j} + b^\beta_{\alpha j} A^*_\beta - c_{\alpha j} A^*_\alpha)\Delta t \right]}{1 - \exp\left[- f(a'_\alpha + b'^\beta_\alpha - c'_\alpha A^*_\alpha)\Delta t \right]} \right\}, \tag{I.5}$$

where $k_{\mathrm{ref},\alpha}$ is the number of bins corresponding to the absolute refractory period of that population. The terms there are given by

$$a_{\alpha j} = e^{-(j-k_{\mathrm{ref},\alpha}+1)\Delta t/\tau_{m,\alpha}}(u_{r,\alpha} - u_{rest,\alpha}) + u_{rest,\alpha} - u_{th,\alpha} - \theta_{\alpha j}, \tag{I.6}$$

$$b^\beta_{\alpha j} = (1 - e^{-(j-k_{\mathrm{ref},\alpha}+1)\Delta t/\tau_{m,\alpha}}) \frac{1 - e^{-\Delta t/\tau_{m,\beta}}}{1 - e^{-\Delta t/\tau_{m,\alpha}}} \tau_m p^\beta_\alpha N^\beta w^\beta_\alpha, \tag{I.7}$$

$$c_{\alpha j} = J_{\theta,\alpha} e^{-T/\tau_{\theta,\alpha}} + \Delta t \sum_{j'=j+1}^{K} \tilde{\theta}_{\alpha j'}, \tag{I.8}$$

$$a'_\alpha = u_{rest,\alpha} - u_{th,\alpha}, \tag{I.9}$$

$$b'^\beta_\alpha = (1 - e^{-\Delta t/\tau_m})\tau_m p^\beta_\alpha N^\beta w^\beta_\alpha, \tag{I.10}$$

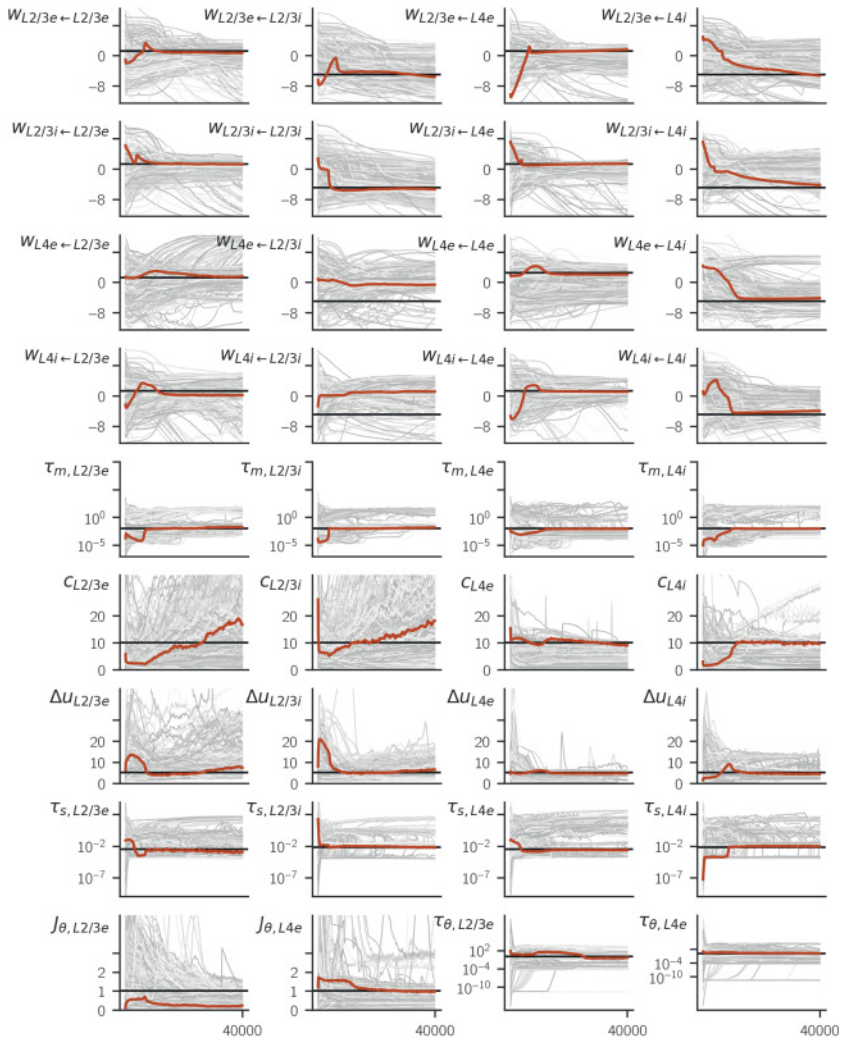

Figure 13: Fit dynamics for the four-population model. Shown are the 622 fits used to infer parameters for the four-population model in section 2.6. Although certain parameters would benefit from more iterations (e.g., c), most have converged within 4×10^4 iterations. Thirty-six parameters were inferred; black lines indicate theoretical values.

$$c'_\alpha = J_{\theta,\alpha} e^{-T/\tau_{\theta,\alpha}}, \tag{I.11}$$

and the inverse link function f and the kernels θ and $\tilde{\theta}$ are as in appendix E.

Equation I.5 can be solved numerically for A^*, after which the other state variables are easily calculated from the expressions in appendix E. We used SciPy's (Jones, Oliphant, & Peterson, 2001) `root` function with an initial guess of $A^*_\alpha = 1$ to solve for A^*. Since the stationary initialization was ultimately only used this to validate algorithm 1 (see appendix C), we did no further analysis of equation I.5, and in particular leave the determination of conditions for which its solutions are unique to future work.

Acknowledgments

We thank the reviewers for improving the manuscript through many valuable comments. We are especially grateful to Tilo Schwalger for his critical comments prior to submission. We also thank Pedro Gonçalves, Giacomo Bassetto, and David Dahmen for further comments and discussions. A.R. and A.L. were supported by NSERC (Canada); A.L. also acknowledges support from the Humboldt Foundation. J.H.M. was supported by the German Research Foundation (DFG) through SFB 1089 and the German Federal Ministry of Education and Research (BMBF, project ADMIMEM, FKZ 01IS18052 A-D).

References

Augustin, M., Ladenbauer, J., Baumann, F., & Obermayer, K. (2017). Low-dimensional spike rate models derived from networks of adaptive integrate-and-fire neurons: Comparison and implementation. *PLOS Computational Biology*, *13*(6), e1005545.

Barak, O. (2017). Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, *46*, 1–6.

Betancourt, M. J., & Girolami, M. (2013). *Hamiltonian Monte Carlo for hierarchical models*. arXiv:1312.0906.

Chizhov, A. V., & Graham, L. J. (2008). Efficient evaluation of neuron populations receiving colored-noise current based on a refractory density method. *Physical Review E*, *77*(1), 011910.

Cunningham, J. P., & Yu, B. M. (2014). Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, *17*(11), 1500–1509.

Doiron, B., Litwin-Kumar, A., Rosenbaum, R., Ocker, G. K., & Josic, K. (2016). The mechanics of state-dependent neural correlations. *Nature Neuroscience*, *19*(3), 383–393.

Dumont, G., Payeur, A., & Longtin, A. (2017). A stochastic-field description of finite-size spiking neural networks. *PLOS Computational Biology*, *13*(8), e1005691.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). *Bayesian data analysis*. Boca Raton, FL: CRC Press.

Gerstner, W. (2000). Population dynamics of spiking neurons: Fast transients, asynchronous states, and locking. *Neural Computation*, *12*(1), 43–89.

Gerstner, W., Paninski, L., Naud, R., & Kistler, W. M. (2014). *Neuronal dynamics from single neurons to networks and models of cognition*. Cambridge: Cambridge University Press.

Girolami, M., & Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *73*(2), 123–214.

Goldwyn, J. H., & Shea-Brown, E. (2011). The what and where of adding channel noise to the Hodgkin-Huxley equations. *PLOS Computational Biology*, *7*(11), e1002247.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA: MIT Press.

Greenberg, D., Nonnenmacher, M., & Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In *Proceedings of the International Conference on Machine Learning. PMLR*, *97*, 2404–2414

Haviv, D., Rivkind, A., & Barak, O. (2019). *Understanding and controlling memory in recurrent neural networks*. arXiv:1902.07275.

Hawrylycz, M., Anastassiou, C., Arkhipov, A., Berg, J., Buice, M., Cain, N., Mind-Scope, (2016). Inferring cortical function in the mouse visual system through large-scale systems neuroscience. *Proceedings of the National Academy of Sciences*, *113*(27), 7337–7344.

Hoffman, M. D., & Gelman, A. (2014). The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, *15*(1), 1593–1623.

Horsthemke, W., & Lefever, R. (2006). *Noise-induced transitions: Theory and applications in physics, chemistry, and biology* (2nd ed.). Berlin: Springer.

Iolov, A., Ditlevsen, S., & Longtin, A. (2017). Optimal design for estimation in diffusion processes from first hitting times. *SIAM/ASA Journal on Uncertainty Quantification*, *5*, 88–110.

Jones, E., Oliphant, T., & Peterson, P. (2001). *SciPy: Open source scientific tools for Python*.

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv:1412.6980.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., & Blei, D. M. (2017). Automatic differentiation variational inference. *Journal of Machine Learning Research*, *18*(14), 1–45.

Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., & Macke, J. H. (2017). Flexible statistical inference for mechanistic models of neural dynamics. In I. Guyon, U. V. Luxburg, S. Gengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*, *30* (pp. 1289–1299). Red Hook, NY: Curran.

Macke, J. H., Buesing, L., Cunningham, J. P., Yu, B. M., Shenoy, K. V., & Sahani, M. (2011). Empirical models of spiking in neural populations. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *24* (pp. 1350–1358). Red Hook, NY: Curran.

Martí, D., Brunel, N., & Ostojic, S. (2018). Correlations between synapses in pairs of neurons slow down dynamics in randomly connected neural networks. *Physical Review E*, *97*(6), 062314.

Mena, G., & Paninski, L. (2014). On quadrature methods for refractory point process likelihoods. *Neural Computation*, *26*(12), 2790–2797.

Mensi, S., Naud, R., Pozzorini, C., Avermann, M., Petersen, C. C. H., & Gerstner, W. (2012). Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms. *Journal of Neurophysiology*, *107*(6), 1756–1775.

Meyer, A. F., Williamson, R. S., Linden, J. F., & Sahani, M. (2017). Models of neuronal stimulus-response functions: Elaboration, estimation, and evaluation. *Frontiers in Systems Neuroscience*, *10*, 109.

Moral, P. D., Doucet, A., & Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *68*(3), 411–436.

Naud, R., & Gerstner, W. (2012). Coding and decoding with adapting neurons: A population approach to the peri-stimulus time histogram. *PLOS Computational Biology*, *8*(10), e1002711.

Neal, R. M. (2012). *MCMC using Hamiltonian dynamics*. arXiv:1206.1901.

Nykamp, D. Q., & Tranchina, D. (2000). A population density approach that facilitates large-scale modeling of neural networks: Analysis and an application to orientation tuning. *Journal of Computational Neuroscience*, *8*(1), 19–50.

Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., & Sussillo, D. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, *15*(10), 805.

Paninski, L., Pillow, J. W., & Simoncelli, E. P. (2004). Maximum likelihood estimation of a stochastic integrate-and-fire neural encoding model. *Neural Computation*, *16*(12), 2533–2561.

Papamakarios, G., & Murray, I. (2016). Fast *o*-free inference of simulation models with Bayesian conditional density estimation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*, *29* (pp. 1028–1036). Red Hook, NY: Curran.

Papamakarios, G., Sterratt, D. C., & Murray, I. (2018). *Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows*. arXiv:1805.07226.

Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E. J., & Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, *454*(7207), 995–999.

Potjans, T. C., & Diesmann, M. (2014). The cell-type specific cortical microcircuit: Relating structure and activity in a full-scale spiking network model. *Cerebral Cortex*, *24*(3), 785–806.

Ramirez, A. D., & Paninski, L. (2014). Fast inference in generalized linear models via expected log-likelihoods. *Journal of Computational Neuroscience*, *36*(2), 215–234.

Rule, M. E., Schnoerr, D., Hennig, M. H., & Sanguinetti, G. (2019). Neural field models for latent state inference: Application to large-scale neuronal recordings. *PLOS Computational Biology*, *15*(11), e1007442.

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *Peer J. Computer Science*, *2*, e55.

Schwalger, T., & Chizhov, A. V. (2019). Mind the last spike-firing rate models for mesoscopic populations of spiking neurons. *Current Opinion in Neurobiology*, *58*, 155–166.

Schwalger, T., Deger, M., & Gerstner, W. (2017). Towards a theory of cortical columns: From spiking neurons to interacting neural populations of finite size. *PLOS Computational Biology*, *13*(4), e1005507.

Skilling, J. (2006). Nested sampling for general Bayesian computation. *Bayesian Analysis*, *1*(4), 833–859.

Sussillo, D., & Barak, O. (2012). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, *25*(3), 626–649.

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., & Gelman, A. (2018). *Validating Bayesian inference algorithms with simulation based calibration*. arXiv:1804.06788.

Team, T. T. D., Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., & Zhang, Y. (2016). *Theano: A Python framework for fast computation of mathematical expressions*. arXiv:1605.02688.

van Haasteren, R. (2014). Marginal likelihood calculation with MCMC methods. In R. van Haasteren (Ed.), *Gravitational wave detection and data analysis for pulsar timing arrays* (pp. 99–120). Berlin: Springer-Verlag.

Vogels, T. P., Rajan, K., & Abbott, L. F. (2005). Neural network dynamics. *Annual Review of Neuroscience*, *28*(1), 357–376.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *37*(3), 328–339.

Wallace, E., Benayoun, M., van Drongelen, W., & Cowan, J. D. (2011). Emergent oscillations in networks of stochastic spiking neurons. *PLOS One*, *6*(5), e14804.

Wilson, H. R., & Cowan, J. D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, *12*(1), 1–24.

Wood, S. N. (2010). Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, *466*(7310), 1102–1104.

Zhao, Y., & Park, I. M. (2016). Interpretable nonlinear dynamic modeling of neural trajectories. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems, 29* (pp. 3333–3341). Red Hook, NY: Curran.

Received October 28, 2019; accepted March 15, 2020.