Boosting the Speed and Precision of Spiking Neural Network Parameter Estimation

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

If we could estimate the parameters of spiking neural networks (SNNs) using limited experimental data, it would greatly enhance our ability to analyze cortical circuits. An effective model for such parameter estimation should be rooted in a microscopic model – a neuron-based, bottom-up approach. Despite having lower complexity compared to a microscopic model, the computational efficiency constraints often make it challenging to apply these models to large-scale SNN parameter estimation. In this study, we employ a multi-population mesoscopic model founded on mean-field and quasi-renewal approximations, which interprets neuronal activity in a more digestible manner and in a lower-dimensional space. By varying various parameters within the populations, we gather corresponding neuronal activity data. Subsequently, we train a simple yet efficient deep learning network to reverse-engineer the SNN parameters from the neuronal activity information. The algorithm we have developed not only significantly accelerates the speed of parameter estimation, but it also successfully circumvents local optima and ensure high performance. Consequently, it can be utilized to simulate and predict various neuronal activities, analyze neuronal dynamics, and could potentially be extended to model the cerebral cortex and analyze neurological disorders.

1 Introduction

Thanks to advancements in large-scale electrophysiological recording technologies [1], we are now capable of recording spiking activity from an increasing number of neurons. This enables us to consider reverse-engineering neuronal parameters from limited experimental data and using these parameters to simulate neural networks in specific brain regions, including both observed and unobserved neurons. However, given the immense number of neurons in the brain, even the significant amount of data we can record represents only a small fraction of the total, which poses considerable challenges for parameter estimation [2, 3].

In reality, a large proportion of neurons can be viewed as belonging to several groups, each exhibiting similar characteristics. In studies simulating cortical networks with spiking neurons, the number of different cell types or neuronal populations in each cortical column ranges from about 10 to 200, yet a single cortical column may contain tens of thousands of simulated neurons, and multiple cortical columns can have millions of neurons [4, 5, 6, 7]. Therefore, a complete microscopic model simulation of neurons is not necessary. The computational efficiency of microscopic models is directly related to the number of neurons, which significantly decreases computation speed. Mesoscopic models, based on effective statistical methods, can better simulate neuronal activity. As mesoscopic models are derived from microscopic models, they can achieve a match with microscopic models while providing better interpretability. Other models at coarser scales, such as neural mass models [8, 9, 10], can describe certain characteristics well, such as how stationary states can match the gain function of a single neuron [11, 12], but they fail to align well with microscopic models in dynamic situations [13, 14].

Given the limited number of neuronal groups, mesoscopic models are constructed from microscopic models, utilizing mean-field neuronal population equations [7, 12, 14, 15] to create low-dimensional large-scale neuronal models. These models can also include the adaptation of neuronal firing voltage thresholds via the quasi-renewal approximation [7]. Then, for estimating network parameters with a large number of hidden neurons, The authors of [3] tackle this problem using gradient descent, a method that, while effective, has certain limitations. It can frequently converge to locally optimal points and requires substantial computational time, despite the relative efficiency of the mesoscopic model. As a result, their work only considers three neural populations. Additionally, for simplification, neuronal threshold voltage adaptation is not taken into account, with only refractoriness considered. Some other works have been proposed to address the challenge in fitting mechanistic spiking network models to empirical data at the mesoscopic scale, including in [16], where a mechanistic but low dimensional and statistically tractable model is applied, enabling the fitting of a mechanistically interpretable representation of neural dynamics.

In this study, we leverage a mesoscopic model derived from a microscopic model based on generalized integrate-and-fire (GIF) models [12, 17, 18]. This model efficiently describes multi-population spiking neural networks to collect extensive neuronal activity data, while considering neuronal threshold voltage adaptation. Compared to microscopic models, the mesoscopic model can gather large amounts of neuronal data at a faster speed, an advantage that becomes more pronounced as the number of neurons increases. We fix certain network parameters and allow key neuronal parameters (including the connectivity matrix – synaptic weights between each pair of populations (J_{syn}), resting potential (μ, which is the sum of resting potential and the external input because their effect on neuronal activities is the same), membrane time constant (τ_m), baseline threshold voltage (u_{th}), adaptation strength (J_θ), and adaptation time scale (τ_θ)) to be randomly sampled within a certain range. This creates a one-to-one correspondence between parameters and neural activity data. Consequently, we use the neural activity data as input and the parameters as labels to train an artificial neural network for efficient spiking neural network parameter estimation. We opt for the relatively simpler gated recurrent unit (GRU) over long short-term memory (LSTM) in a recurrent neural network (RNN) [19, 20] to implement inverse parameter estimation. It reads temporal activity data from multiple populations and ultimately outputs the six types of parameters mentioned above.

2 Background: mesoscopic modeling

The simplicity of neurons is crucial in simulating neural activity due to the vast number of neurons in the brain. Biophysical neuron models can be effectively approximated using simplified spiking neurons, disregarding intricate details such as dendritic nonlinearity [21, 22, 23]. These simplified models, such as noisy leaky integrate-and-fire (LIF) neurons, provide a practical approach to simulate neuronal behavior and introduce the concept of renewal-type neurons [12, 14]. Renewal-type neurons are characterized by their limited memory of spike history. The mean-field description, which considers interactions among multiple populations of renewal-type neurons, can be accurately represented by an integral equation [12, 14, 24, 25, 26]. In [7], the authors present a mesoscopic-level stochastic integral equation.

Mesoscopic modeling used in this work relies heavily on the above two key approximations: the mean-field approximation and the quasi-renewal approximation, and take the threshold adaptation into account. At the heart of the mean-field approximation lies the construct of a neural network composed of multiple populations. Each population is an aggregate of identical neurons that are probabilistically interconnected, forming what can effectively be considered a local network. The key assumption is the representability of a neuron population's collective behavior by the mean activity [7, 12, 14, 24, 25, 26]. This critical abstraction simplifies the intricacies inherent in individual neuronal dynamics, thus enabling tractable yet substantive low-dimensional simulations of neural activity. The quasi-renewal approximation leverages GIF neurons to postulate that the effective threshold depends solely on the last spike time, utilizing a statistical kernel to encapsulate the history of population activity [7, 12, 14, 27]. This approximation, at its core, temporally distills the statistical attributes of the neural population into mesoscopic variables. In this context, the mesoscopic model can be thought of as an algorithm that employs mesoscopic variables as effective intermediary entities

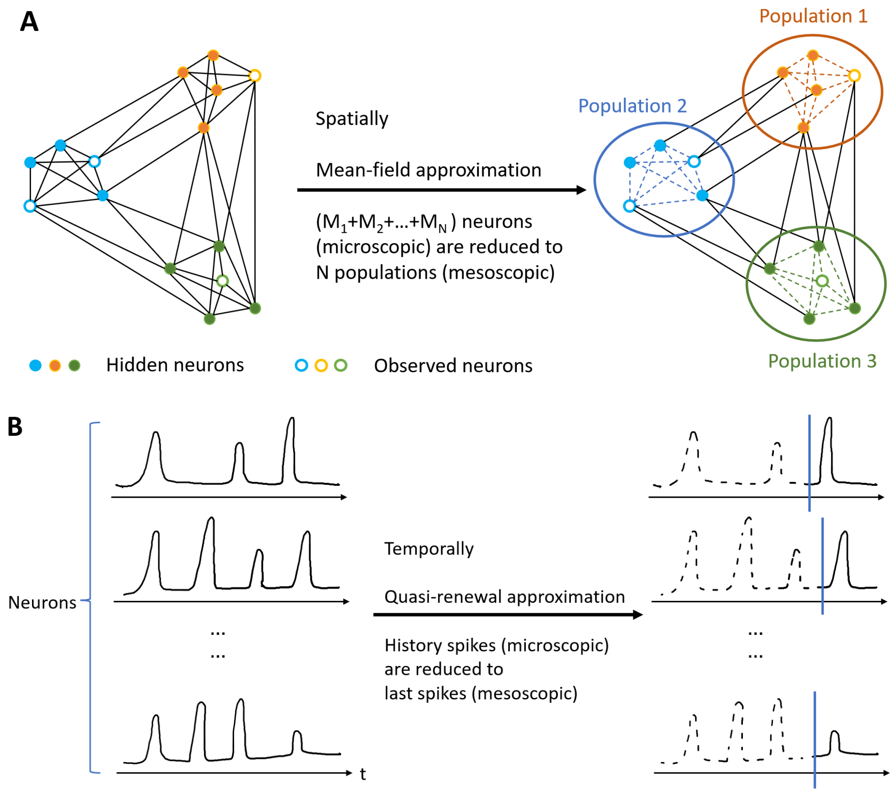
A
Spatially
Mean-field approximation
(M1+M2+...+MN) neurons (microscopic) are reduced to N populations (mesoscopic)
Population 1
Population 2
Population 3
Hidden neurons
Observed neurons
B
Neurons
Temporally
Quasi-renewal approximation
History spikes (microscopic) are reduced to last spikes (mesoscopic)
t
t

Figure 1: **Schematic diagram of mean-field and quasi-renewal approximation. (A)** The mean-field approximation treats homogeneous neurons as a collective, reducing the dimensionality of the variable from the number of individual neurons to the number of populations. Different colors represent neurons with distinct properties, which are grouped accordingly. While connections within groups are still taken into account, they are represented probabilistically, with a primary focus on connections between groups. **(B)** The quasi-renewal approximation focuses on the most recent spikes and describes the historical spikes using statistical kernels. By simplifying the temporal complexity and capturing mesoscopic phenomena from a microscopic level, this approximation provides a more concise representation.

from microscopic but not at a over-coarse level, thereby facilitating a more efficient portrayal of neural activity.

For a network that can be approximated by multiple populations, considering a population in them, which is with N interconnected neurons of the same type, use p^{ξ} to denote its probability of connecting a neuron from another population ξ. Use $s_i(t)$ to denote the spikes of a neuron i in the population as time goes forward, $t_{i,k}$ is the time of the k-th spike and δ is a spike – the Dirac function. Let A denote the population activity and $\bar{A}$ denote the mean population activity, and $\Delta n(t_1)$ be the firing number of neurons in the population in time t_1 (after time discretization, t_1 represents a small time period Δt). Then we have the following expression change from single neurons to a population (Equation 1).

$$s_i(t) = \sum_k \delta\left(t - t_{i,k}\right) \rightarrow A\left(t_1\right) = \frac{\Delta n\left(t_1\right)}{N\Delta t}, \bar{A}\left(t_1\right) = \frac{\Delta \bar{n}\left(t_1\right)}{N\Delta t} \tag{1}$$

Adopting mean-field approximation, the current and membrane potential can be shown as Equation 2 and Equation 3.

$$RI_i(t) = \tau_{\mathrm{m}} \sum_{\xi=1}^{X} p^{\xi} N^{\xi} w^{\xi} \left(\epsilon^{\xi} * A^{\xi}\right)(t) \tag{2}$$

$$\tau_{\mathrm{m}} \frac{\partial u}{\partial t} = -u + \mu(t) + \tau_{\mathrm{m}} \sum_{\xi=1}^{X} p^{\xi} N^{\xi} w^{\xi} \left(\epsilon^{\xi} * A^{\xi}\right)(t) \tag{3}$$

where R and τ_{m} are the membrane resistance and time constant, respectively, and w^{ξ} (mV) are the synaptic weights between the current population and population ξ, and $\epsilon^{\xi}(t)$ is the normalized synaptic kernel, representing the postsynaptic current. $\mu(t)$ is the sum of constant external input (e.g., step input) and the resting potential, which is a key parameter we need to predict when estimating, so is τ_{m}. Another key parameter here is the weights between every two population form a square connectivity matrix J_{syn} whose dimension is the number of populations.

For quasi-renewal approximation, based on the GIF conditional intensity $\lambda_i(t) = f\left(u_i(t) - \vartheta_i(t)\right)$ ($\vartheta_i(t)$ is the threshold), the firing rate $\lambda(t)$ in the population can be rewrote in a simple but effective format – $\lambda_i\left(t \mid \hat{t}_i\right)$ that only considers the last spike $\hat{t}_i$ of the neuron and the history of the population activity. Further, in the reduced $\lambda_i(t) \approx f\left(u_A\left(t, \hat{t}_i\right) - \vartheta_i(t)\right)$, the threshold $\vartheta_i(t)$ is still determined by all the spikes in the firing history, which is not a renewal type – it is not simply time dependent. The quasi-renewal approximation uses a adaptation kernel to describe the effect of spikes before the last spike, which can be regarded as the "average" of those spikes (the spike activities of neurons correspond to an inhomogeneous Poisson process). Then we have $\lambda_i(t) \approx f\left(u_A\left(t, \hat{t}_i\right) - \vartheta_A\left(t, \hat{t}_i\right)\right)$, which is denoted as $\lambda_A\left(t \mid \hat{t}_i\right)$, where $\vartheta_A(t, \hat{t}) = u_{\mathrm{th}} + \theta(t - \hat{t}) + \int_{-\infty}^{\hat{t}} \tilde{\theta}\left(t - t'\right) A\left(t'\right) \mathrm{d}t'$ and $\tilde{\theta}(t) = \Delta_u\left[1 - e^{-\theta(t)/\Delta_u}\right]$. Δ_u is the softness coefficient. u_{th} is the potential threshold – a key parameter that needs to be predicted. $\theta(t)$ here is the basic kernel of the adaptation of spikes on the potential threshold, which is $\theta(t) = \frac{J_\theta}{\tau_\theta} e^{-t/\tau_\theta}$, where $\frac{1}{\tau_\theta} e^{-t/\tau_\theta}$ is a normalized term with area under the curve being 1, and J_θ and τ_θ are adaptation strength and adaptation time scale, respectively (both are key parameters for estimation).

By introducing mesoscopic variables from the microscopic level, we reach to a point with a low-dimensional variable space but not a over-coarse level. Finally, after some statistical derivation such as from approximation from binomial distribution to Poisson distribution, from a sufficient number of Poisson distribution to Guassian distribution and transformation of statistical characteristics (for detailed derivations, please refer to the references mentioned in the beginning part of this section), we have:

$$A(t) = \bar{A}(t) + \sqrt{\frac{\bar{A}(t)}{N}} \zeta(t) \tag{4}$$

where $\zeta(t)$ is a Gaussian white noise, and

$$\bar{A}(t) = \int_{-\infty}^{t} \lambda_A(t \mid \hat{t}) S(t \mid \hat{t}) A(\hat{t}) \mathrm{d}\hat{t} + \Lambda(t)\left(1 - \int_{-\infty}^{t} S(t \mid \hat{t}) A(\hat{t}) \mathrm{d}\hat{t}\right) \tag{5}$$

$$\lambda_A(t \mid \hat{t}) = c \exp\left(\frac{u(t, \hat{t}) - \vartheta_A(t, \hat{t})}{\Delta_u}\right), \quad \Lambda(t) = \frac{\int_{-\infty}^{t} \lambda_A(t \mid \hat{t}) v(t, \hat{t}) \mathrm{d}\hat{t}}{\int_{-\infty}^{t} v(t, \hat{t}) \mathrm{d}\hat{t}} \tag{6}$$

where $S\left(t_1 \mid t_2\right) = \frac{\langle \hat{m}(t_1, t_2) \rangle}{\Delta n\left(t_2\right)}$ is the survival rate, and $v\left(t_l, t_k\right) = \frac{\left\langle \Delta \hat{m}^2(t_1, t_2) \right\rangle}{N \Delta t}$ is a variance, and c is a constant in the conditional intensity. $m\left(t_1, t_2\right)$ is the number of neurons at time t_1 with their last spikes at time t_2, with $\sum_{k=-\infty}^{t_{l-1}} m\left(t_1, t_2\right) = N$, and the mean $\langle \hat{m}(t_1, t_2) \rangle$ is a mesoscopic variable. Variables should be given initial conditions as a start of the simulation of the mesoscopic model.

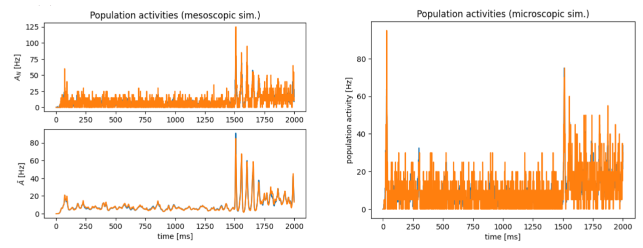

Figure 2: **Neural activities match between the mesoscopic and microscopic models (2-pop) (Left)** Mean neural activity and random neural activity of the mesoscopic model. **(Right)** Random neural activity of microscopic models

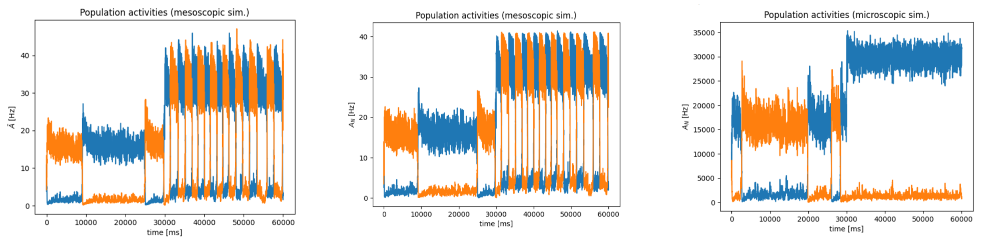

Figure 3: **Neural activities match between the mesoscopic and microscopic models (3-pop) (Left)** Mean neural activity of the mesoscopic model. **(Middle)** Random neural activity of the mesoscopic model. **(Right)** Random neural activity of microscopic models.

Now, we have provided a concise overview of the entire process, encompassing the transition from the microscopic to the mesoscopic scale. Additionally, we have elucidated the key parameters involved and outlined the methodology for obtaining mesoscopic neuron activity. In the real-time simulation, we will discretize population equations considering the activity in each time bin.

3 Parameter estimation

3.1 Reproducing and comparing the mesoscopic and microscopic models

We employ a network composed of two distinct populations, each housing a different number of neurons: 800 in the excitatory population and 200 in the inhibitory population to validate the similarity between two types of models. This structure is both simulated using the microscopic model and the mesoscopic model. The resulting data from these simulations show strikingly similar outcomes with adaptation taken into account. The close alignment between the results produced by the two models highlights the mesoscopic model's capability to accurately and efficiently replicate the results of its microscopic counterpart. This further reinforces the mesoscopic model's suitability for analyzing large-scale networks without sacrificing the detailed dynamics captured in the microscopic model. See Figure 2. For a 3-population (400 in the excitatory population, 200 in the inhibitory population, and another 400 in a second excitatory population; only two are shown in the plotted figures considering the competing – the winner takes all, same as the estimated case), we can see the bistable property under three populations, and also a good match between two levels' models. See Figure 3.

However, the execution speed of the mesoscopic model significantly outperforms that of the microscopic model. This speed advantage becomes particularly evident when dealing with networks that encompass a large number of neurons. The increased efficiency of the mesoscopic model in these situations allows for more rapid data collection and processing, making it an ideal tool for large-scale neural network simulations. For larger population of neurons, such as shown in Table 2, the microscopic model takes hours or even longer for once simulation, while the mesoscopic model only needs less than 1 minute to run for a population number less than 10.

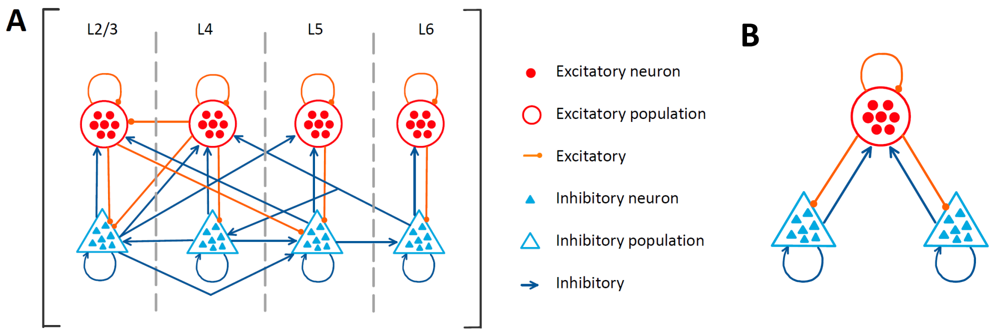

Figure 4: **Schematic of network architecture.** **(A)** The architecture of the eight-population is composed of four layers (L2/3, L4, L5, and L6). Each layer comprises both excitatory and inhibitory populations. **(B)** The architecture of the three-population comprises two competing excitatory populations and a shared inhibitory population, representing a winner-take-all SNN.

3.2 Parameter settings and data collection

The different schematic of network architecture of the eight-population and the three-population are shown in Figure 4.

Let's denote the number of populations by Y and the number of exponential kernels used to approximate the adaptation from the spikes before the last spike by Z. Consequently, the dimensions of the six parameters, connectivity matrix J_{syn}, resting potential μ, membrane time constant τ_m, baseline threshold voltage u_{th}, adaptation strength J_θ, and adaptation time scale τ_θ, are $Y \times Y$, Y, Y, Y, $Y \times Z$ and $Y \times Z$ respectively.

During the data normalization process, after collecting a certain amount of data, we identify the number with the largest absolute value among the collected data. Subsequently, we divide all the data by this maximum value, resulting in normalized values ranging between -1 and 1 (0 and 1 if all positive).

On the output side, the activity of neurons, denoted by A, represents the outcome. For the 8-population network, this refers to the activity of eight neurons, resulting in an eight-dimensional output or Y, corresponding to the population number. Additionally, since the activity has been observed over T time points, the dimension of matrix A is $Y \times T$. To normalize matrix A, we traverse all the values. For each population, we calculate the maximum value across all instances of A, and then divide all the values within that population by its corresponding maximum value, ensuring normalization.

Apart from the mentioned six key neuronal parameters (including synaptic weights between each pair of populations, resting potential, membrane time constant, baseline threshold voltage, adaptation strength, and time scale) for estimation, we fix other parameters (see Table 1 and Table 2 for the examples of 8 populations), which are for a network of modified Potjans-Diesmann model. We here omit the fixed parameters for 3 populations, which is similar in some values and much simpler in parameters like connectivity matrix.

3.3 Use GRU for estimation

GRU is a type of RNN architecture, simpler than LSTM, which is designed to model sequential data and capture long-term dependencies within the data. GRUs are particularly effective for tasks such as sequence prediction, language modeling, and time series analysis.

The main advantage of using GRU over traditional RNNs is its ability to address the vanishing gradient problem, which is a common issue in training deep neural networks, by introducing gating mechanisms that regulate the flow of information within the network. In the context of identifying neuron activity data, GRUs can be effective because they can capture the temporal dynamics of the activity patterns. Neurons often exhibit complex temporal dependencies, where the current state of a neuron depends not only on its previous states but also on the states of other neurons in the network. GRUs can learn and model these dependencies, making them suitable for tasks such as predicting future neuron activity based on past observations.

Table 1: Connection probability.

	L2/3e	**L2/3i**	**L4e**	**L4i**	**L5e**	**L5i**	**L6e**	**L6i**
L2/3e	0.1009	0.1689	0.0437	0.0818	0.0323	0	0.0076	0
L2/3i	0.1346	0.1371	0.0316	0.0515	0.0755	0	0.0042	0
L4e	0.0077	0.0059	0.0497	0.135	0.0067	0.0003	0.0453	0
L4i	0.0691	0.0029	0.0794	0.1597	0.0033	0	0.1057	0
L5e	0.1004	0.0622	0.0505	0.0057	0.0831	0.3726	0.0204	0
L5i	0.0548	0.0269	0.0257	0.0022	0.06	0.3158	0.0086	0
L6e	0.0156	0.0066	0.0211	0.0166	0.0572	0.0197	0.0396	0.2252
L6i	0.0364	0.001	0.0034	0.0005	0.0277	0.008	0.0658	0.1443

Table 2: Parameters of the modified Potjans-Diesmann model.

population	**L2/3e**	**L2/3i**	**L4e**	**L4i**	**L5e**	**L5i**	**L6e**	**L6i**
synaptic time constants [s]	0.0005							
transmission constant delay [s]	0.0015							
refractory period [s]	0.002							
softness of threshold adaptation [mV]	5.0							
external step input	[0.06s, 0.09s]							
neuron number	20683	5834	21915	5479	4850	1065	14395	2948

By training a GRU model on a dataset of neuron activity data, it can learn the underlying patterns and relationships within the data. This learned model can then be used to make predictions or classify different patterns of neuron activity. We can use arbitrary input length of the neural activity sequence.

Our GRU: The model begins with a GRU layer with three hidden layers and two stacked layers (or changing according to neural populations), and an input size of 8 (if 8 populations). This layer processes the input sequences and provides the output and the final hidden state. The GRU is especially suited for sequence data because it can capture long-term dependencies due to its gating mechanisms.

The output from the GRU is then passed through several fully connected (Linear) layers, each of which maps the GRU output to a different set of neural network parameters. These include synaptic weights (J_{syn}), resting potentials (mu), membrane time constants (tau_m), base threshold potentials (V_{th}), threshold potential adaption strength (J_θ), and threshold potential adaptation time scale (τ_θ). Our GRU model possesses remarkably small weights, with the size of the trained weights being only a few hundred kilobytes. This leads to a fast inference speed.

The use of separate layers for each set of parameters allows the model to learn a distinct function mapping from the input sequences to each parameter set. As a result, this model facilitates the efficient estimation of neural network parameters from sequences of neural activity.

3.4 Resulting activities based on estimation

We compared the activity curves of a specific neuronal population from the original mesoscopic model with those generated by the mesoscopic model using the estimated parameters. The high degree of similarity between these activity curves validates the effectiveness of our proposed parameter estimation method, which leverages both GRU and data from the mesoscopic model. See Figure 6

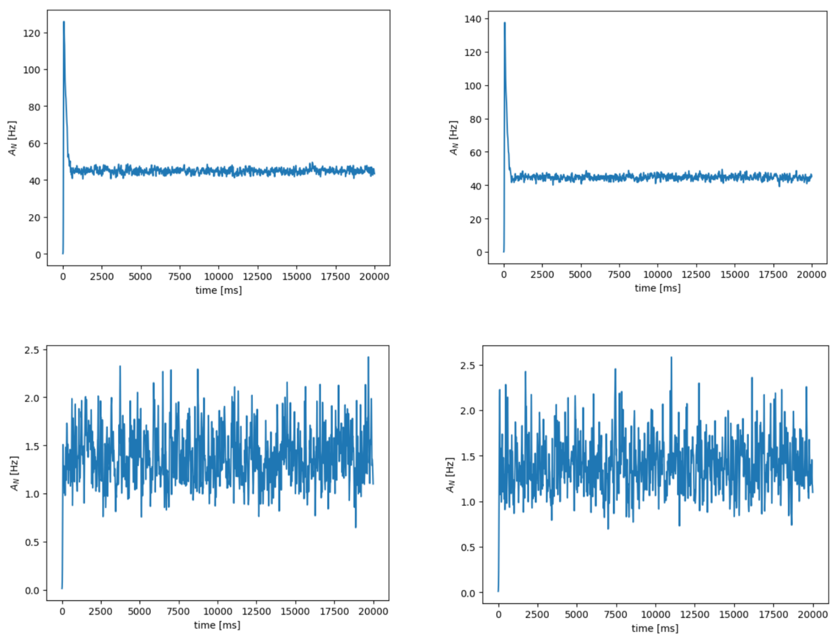

Figure 5: **Neuron activities comparison (3-population case).** 2 were randomly selected from 3 populations for display. The left and right column represents the neuron activities of original model and those generated by the mesoscopic model using the estimated parameters respectively, revealing a similar pattern of results between them.

and Figure 5 for the good match between our estimated model and the original model (labels) in 8 and 3 population scenarios (more results are not included here due to space limit).

4 Conclusion

In conclusion, this research highlights the significance of the mesoscopic model in simulating neuron models and addresses a crucial limitation in parameter estimation by integrating artificial neural networks.

The mesoscopic model, building upon the microscopic model, offers improved statistical properties. Exploiting this advantage, the mesoscopic model becomes a valuable tool for efficiently generating large-scale neuron models, facilitating rapid data collection. While the mesoscopic model excels in simulation capabilities, parameter estimation remains a bottleneck due to its slow process. To overcome this challenge, we have explored the utilization of artificial neural networks as a substitute for traditional parameter estimation methods. Our findings demonstrate that artificial neural networks provide an efficient and high-speed approach to estimate neural network parameters.

By extensively validating numerous mesoscopic models, we have successfully established the effectiveness of our approach. The simulation results obtained from the mesoscopic model closely align with those derived from the microscopic model. This validation process involved the meticulous collection of extensive data from mesoscopic models, which was subsequently employed to perform accurate parameter estimation.

In summary, our research underscores and enhances the utility of the mesoscopic model in neuron modeling. Through the integration of artificial neural networks, we have demonstrated an efficient and effective means of estimating parameters. These advancements contribute significantly to our broader goal of advancing our understanding of neural systems and their underlying dynamics, thereby opening up avenues for further breakthroughs in neuroscience research. (Further details and additional information can be found in the supplementary materials.)

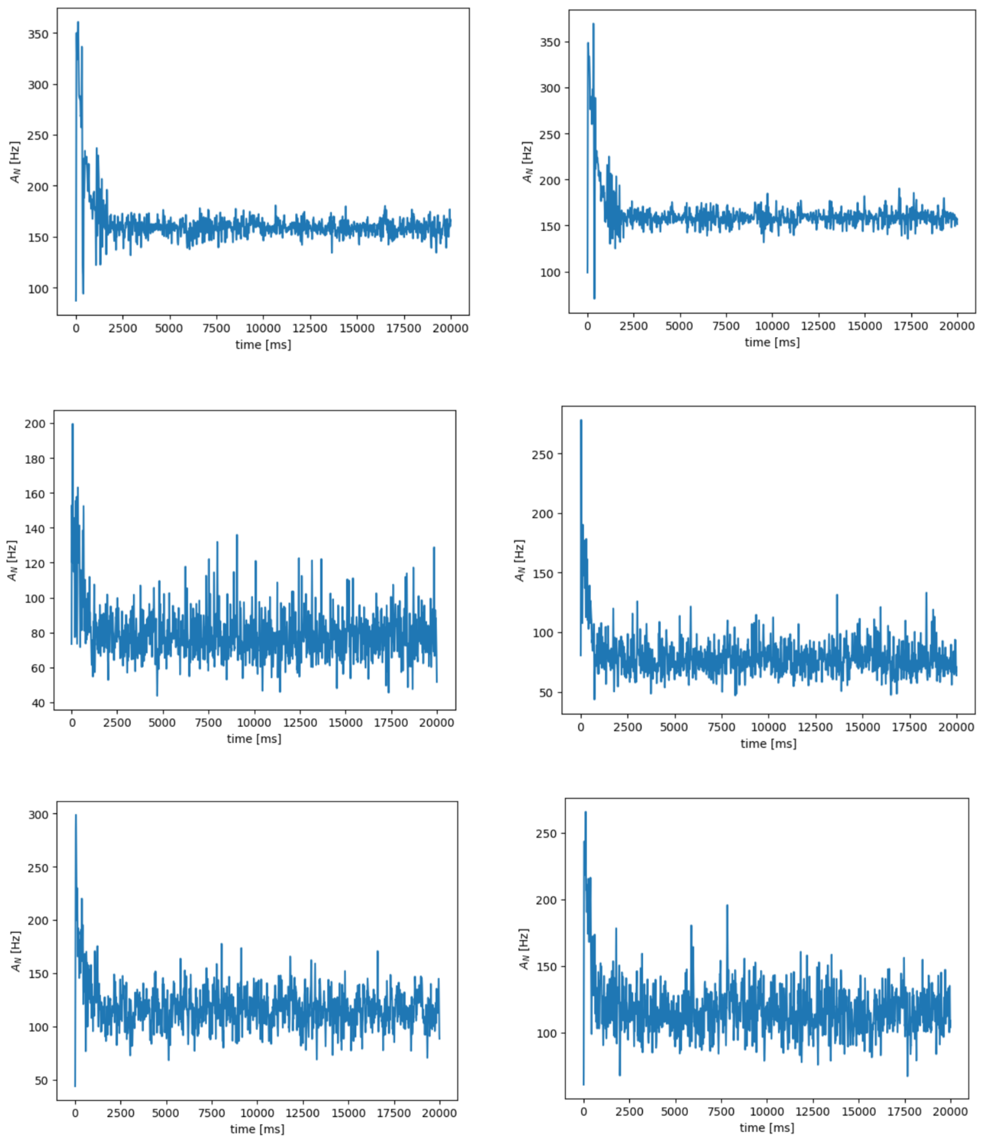

Figure 6: **Neuron activities comparison (8-population case).** 3 were randomly selected from 8 populations for display. The left and right column represents the neuron activities of original model and those generated by the mesoscopic model using the estimated parameters respectively, revealing a similar pattern of results between them.

References

[1] N. A. Steinmetz, C. Koch, K. D. Harris, and M. Carandini, “Challenges and opportunities for large-scale electrophysiology with neuropixels probes,” *Current opinion in neurobiology*, vol. 50, pp. 92–100, 2018.

[2] P. Gao and S. Ganguli, “On simplicity and complexity in the brave new world of large-scale neuroscience,” *Current opinion in neurobiology*, vol. 32, pp. 148–155, 2015.

[3] S. Wang, V. Schmutz, G. Bellec, and W. Gerstner, “Mesoscopic modeling of hidden spiking neurons,” *arXiv preprint arXiv:2205.13493*, 2022.

[4] T. C. Potjans and M. Diesmann, “The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model,” *Cerebral cortex*, vol. 24, no. 3, pp. 785–806, 2014.

[5] H. Markram, E. Muller, S. Ramaswamy, M. W. Reimann, M. Abdellah, C. A. Sanchez, A. Ailamaki, L. Alonso-Nanclares, N. Antille, S. Arsever, *et al.*, “Reconstruction and simulation of neocortical microcircuitry,” *Cell*, vol. 163, no. 2, pp. 456–492, 2015.

[6] E. M. Izhikevich and G. M. Edelman, “Large-scale model of mammalian thalamocortical systems,” *Proceedings of the national academy of sciences*, vol. 105, no. 9, pp. 3593–3598, 2008.

[7] T. Schwalger, M. Deger, and W. Gerstner, “Towards a theory of cortical columns: From spiking neurons to interacting neural populations of finite size,” *PLoS computational biology*, vol. 13, no. 4, p. e1005507, 2017.

[8] W. J. Freeman *et al.*, *Mass action in the nervous system*, vol. 2004. Citeseer, 1975.

[9] O. David and K. J. Friston, “A neural mass model for meg/eeg:: coupling and neuronal dynamics,” *NeuroImage*, vol. 20, no. 3, pp. 1743–1755, 2003.

[10] R. Moran, D. A. Pinotsis, and K. Friston, “Neural masses and fields in dynamic causal modeling,” *Frontiers in computational neuroscience*, vol. 7, p. 57, 2013.

[11] V. K. Jirsa and H. Haken, “A derivation of a macroscopic field theory of the brain from the quasi-microscopic neural dynamics,” *Physica D: Nonlinear Phenomena*, vol. 99, no. 4, pp. 503–526, 1997.

[12] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[13] G. Deco, V. K. Jirsa, P. A. Robinson, M. Breakspear, and K. Friston, “The dynamic brain: from spiking neurons to neural masses and cortical fields,” *PLoS computational biology*, vol. 4, no. 8, p. e1000092, 2008.

[14] W. Gerstner, “Population dynamics of spiking neurons: fast transients, asynchronous states, and locking,” *Neural computation*, vol. 12, no. 1, pp. 43–89, 2000.

[15] N. Brunel and V. Hakim, “Fast global oscillations in networks of integrate-and-fire neurons with low firing rates,” *Neural computation*, vol. 11, no. 7, pp. 1621–1671, 1999.

[16] A. René, A. Longtin, and J. H. Macke, “Inference of a mesoscopic population model from population spike trains,” *Neural computation*, vol. 32, no. 8, pp. 1448–1498, 2020.

[17] W. Gerstner and R. Naud, “How good are neuron models?,” *Science*, vol. 326, no. 5951, pp. 379–380, 2009.

[18] S. Mensi, R. Naud, C. Pozzorini, M. Avermann, C. C. Petersen, and W. Gerstner, “Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms,” *Journal of neurophysiology*, vol. 107, no. 6, pp. 1756–1775, 2012.

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[21] W. M. Kistler, W. Gerstner, and J. L. v. Hemmen, “Reduction of the hodgkin-huxley equations to a single-variable threshold model,” *Neural computation*, vol. 9, no. 5, pp. 1015–1045, 1997.

[22] R. Jolivet, T. J. Lewis, and W. Gerstner, “Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy,” *Journal of neurophysiology*, vol. 92, no. 2, pp. 959–976, 2004.

[23] R. Brette and W. Gerstner, “Adaptive exponential integrate-and-fire model as an effective description of neuronal activity,” *Journal of neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.

[24] W. Gerstner, “Time structure of the activity in neural network models,” *Physical review E*, vol. 51, no. 1, p. 738, 1995.

[25] T. Schwalger and A. V. Chizhov, “Mind the last spike—firing rate models for mesoscopic populations of spiking neurons,” *Current opinion in neurobiology*, vol. 58, pp. 155–166, 2019.

[26] H. R. Wilson and J. D. Cowan, “Excitatory and inhibitory interactions in localized populations of model neurons,” *Biophysical journal*, vol. 12, no. 1, pp. 1–24, 1972.

[27] R. Naud and W. Gerstner, “Coding and decoding with adapting neurons: a population approach to the peri-stimulus time histogram,” 2012.

Appendices of:

Boosting the Speed and Precision of Spiking Neural Network Parameter Estimation

A Microscopic model and mesoscopic model

Our perspective focuses on a network structure containing multiple interacting, homogeneous clusters. In this context, it implies that each cluster contains spiking neurons with similar inherent characteristics and random intra-cluster and inter-cluster connectivity. The cluster is referred as the population. With empirical data and some theoretical reference values, it will provides us with neuron parameters typical to each population, neuron count within each population, and standard connectivity probabilities and intensities both internally and externally [1–9]. Consequently, this allows for the creation of a spiking neural network (SNN) simulation at the cellular scale, through numerical integration of the spiking patterns of every (distinct) neuron – labeled as the 'microscopic' level (the microscopic model). Yet, this level poses challenges due to the high computational cost and the convoluted nature of the complete microscopic network dynamics. Then, with applying the mean-field approach to encapsulate the intermediate ('mesoscopic') dynamics of the intercommunicating groups to counteract these issues, we have the mesoscopic model [10, 11].

Microscopic model Here, we consider a general SNN of generalized integrate-and-fire (GIF) neurons. Compared to the leaky integrate-and-fire (LIF) model, the GIF model includes additional features to make the model more biologically accurate. It accounts for spike-triggered adaptation, where the neuron's response changes based on spike history. This is done by adding extra terms to the differential equation of the LIF model that account for spike-triggered adaptation of a dynamic threshold for firing [12–15] and a escape noise mechanism [2, 14, 16, 17].

When considering a single neuron, we use $s_i(t)$ to denote the spikes of a neuron i as time goes forward, and let $t_{i,k}$ be the time of the k-th spike and δ be the Dirac function representing a spike. We have that the spike train $s_i(t)$ is the sum of a series of shifted Dirac functions $s_i(t) = \sum_k \delta\left(t - t_{i,k}\right)$. The synaptic input current of the neuron can be represented as the aggregate of post-synaptic currents initiated by each pre-synaptic neuron's spike.

$$RI_{\mathrm{syn},i}(t) = \tau_{\mathrm{m}} \sum_j w^{ij} \left(\epsilon^{ij} * s_j\right)(t) \tag{1}$$

where R represents the membrane resistance and τ_{m} denotes the membrane time constant, both associated with neuron i. The synaptic weights between neurons i and j are established by the parameter w^{ij}. A value of zero for w^{ij} signifies the absence of a connection between neurons i and j. Also, we have $w^{ii} = 0$, indicating that a neuron does not form a connection with itself, thus, $j \neq i$. The synaptic kernel $\epsilon^{ij}(t)$ is characterized as the post-synaptic current, normalized by its charge (or, equivalently, its integral). We consider the synaptic kernel as a single exponential of the form $\epsilon^{ij}(t) = \mathcal{H}(t - \Delta^{ij}) e^{-(t-\Delta^{ij})/\tau^j_{\mathrm{syn}}} / \tau^j_{\mathrm{syn}}$ ($1/\tau^j_{\mathrm{syn}}$ normalizes the kernel), where $\mathcal{H}$ is the Heaviside step function, and τ^j_{syn} and Δ^{ij} are the synaptic time constant and the synaptic delay, respectively, both between neurons i and j. The units for w^{ij} are given as mV, while $\epsilon^{ij}(t)$ carries the unit $1/\mathrm{sec}$. The symbol $*$ signifies the convolution operation.

If we consider a network that can be reasonably approximated by multiple neuronal populations, let's take neuron i belonging to one such population. In this scenario, the synaptic input current of neuron i can be recast as the summation of connections originating from all neuronal populations, which also includes the population to which neuron i itself belongs.

$$RI_{\mathrm{syn},i}(t) = \tau_{\mathrm{m}} \sum_{\xi} w^{i\xi} \sum_{j \in \Pi_i^{\xi}} \left(\epsilon^{i\xi} * s_j^{\xi} \right)(t) \tag{2}$$

where ξ represents the index of neuronal populations. The superscript $i\xi$ signifies the relationship between neuron i and neurons belonging to population ξ, wherein neurons within a population are assumed to have identical parameters. Π_i^{ξ} symbolizes the set of neurons that are part of population ξ and are interconnected with neuron i. In Equation 1, the indices i and j cannot be the same, or $w^{ii} = 0$, implying that a neuron cannot connect with itself. However, in Equation 2, population ξ can indeed be the same population in which neuron i resides. This indicates that the connections within a population (from a microscopic perspective, it is equivalent to connecting a neuron not within the same population). However, if neuron i is part of population ξ, then the set Π_i^{ξ} must not include neuron i.

The GIF model incorporates a conditional intensity function $\lambda_i(t) = f_i\left(u_i(t) - \vartheta_i(t)\right)$ to capture the probability of neuron firing given a specific threshold and membrane potential, both of which are significant parameters in this model. Here, $\vartheta_i(t)$ represents the threshold. The function $f_i(x)$ (the exponential link function) is defined as $f_i(x) = c_i e^{x/\Delta_{u,i}}$. In this context, c_i is the escape rate at the threshold (the base rate of the exponential link function), and $\Delta_{u,i} > 0$ illustrates the degree of threshold softness. Given $\lambda_i(t)$, we can have the firing probability of neuron i at time t:

$$P_i(t) = 1 - e^{-\int_t^{t+\Delta t} \lambda_i(\tau)\mathrm{d}\tau} \approx 1 - e^{-\bar{\lambda}_i(t)\Delta t} \approx \bar{\lambda}_i \Delta t \tag{3}$$

where $P_i(t) = 1 - e^{-\int_{t_l}^{t_l+\Delta t} \lambda_i(t)\mathrm{d}t}$ is held as valid since $\lambda_i(t)$ is essentially the rate of a Poisson distribution for the firing of neuron i. We can readily deduce that the probability of x events occurring in a unit time is given by $P\{X = x\} = \frac{\lambda^x e^{-\lambda}}{x!}$. Extending this to x events in time Δt, we have $P\{X = x, \lambda\Delta t\} = \frac{(\lambda\Delta t)^x e^{-\lambda\Delta t}}{x!}$. Consequently, the probability of no events occurring within the time interval $[t, t + \Delta t)$ can be written as $P\{X = 0, \lambda\Delta t\} = e^{-\lambda\Delta t}$. As such, if we consider an inhomogeneous Poisson process, the probability of observing no events within the time interval $[t, t + \Delta t)$ is represented by the second term in Equation 3. The complementary event, signified by subtracting this probability from 1, represents the probability of at least one spike occurring in the interval. Indeed, the approximation assumes that Δt is sufficiently small such that the probability of multiple spikes occurring within this interval becomes negligibly small. As a result, these events can safely be disregarded when considering the model's behavior.

The escape rate $\lambda_i(t)$ needs to work with the potential from the subthreshold dynamics:

$$\tau_{\mathrm{m}} \frac{\mathrm{d}u_i}{\mathrm{d}t} = -u_i + \mu_i(t) + RI_{\mathrm{syn},i}(t) \tag{4}$$

where $\mu_i(t) = u_{\mathrm{rest},i} + RI_{\mathrm{ext},i}(t)$ represents the convergent value of the potential when there is no synaptic current input, composed of a static resting potential $u_{\mathrm{rest},i}$ and an external stimulus $I_{\mathrm{ext},i}$. When considering threshold adaptation, every spike $t_{i,k}$ contributes to the dynamic threshold $\vartheta_i(t)$ through the spike-triggered threshold kernel $\theta_i\left(t - t_{i,k}\right)$. Thus, $\vartheta_i(t)$ is represented as $u_{\mathrm{th},i} + \sum_{t_{i,j}<t} \theta_i\left(t - t_{i,k}\right) = u_{\mathrm{th},i} + (\theta_i * s_i)(t)$, where the convolution is $(\theta_i * s_i)(t) = \int_{-\infty}^{t} \theta_i(t - \tau)\, s_i(\tau)\,\mathrm{d}\tau$.

If we prefer to avoid the manual reset of the membrane potential after firing required by the LIF model (Equation 4 only holds for subthreshold dynamics), we can explore a variant known as the spike-response model or the generalized linear model (GLM) [14, 18–22]. This model introduces a spike-triggered potential kernel $\eta_i(t)$, akin to the spike-triggered threshold kernel, to accommodate effects, including the refractoriness induced by prior spikes. Fortunately, $\eta_i(t)$ can be integrated into the dynamic threshold $\theta_i(t)$, that is, $\theta \to (\theta - \eta)$ and $\eta \to 0$. The resulting membrane potential then acts as a free membrane potential, unaffected by any previous spike history. Therefore, the final mathematical form is the same.

Mesoscopic model Here, we transition from the above microscopic model to a neuronally-grounded mesoscopic model, shifting our perspective from the level of individual neurons to the aggregate level of neuronal populations and thereby effectively reducing the statistical dimensionality and computational complexity of the model [10, 11, 23].

In a network that can be approximated by several populations, consider a population consisting of N interconnected neurons of the same type. Let p^{ξ} represents the probability of this population connecting to a neuron from a population ξ. The population activity is represented by A and its mean population activity by $\bar{A}$. The number of neuron firings in the population during a time period t is represented by $\Delta n(t)$. After discretizing time (according to [14], transitioning from discrete to continuous time is feasible), t now stands for a small time interval Δt. Hence, the population activity can be represented as follows.

$$A\left(t\right)=\frac{\Delta n\left(t\right)}{N\Delta t},\ \bar{A}\left(t\right)=\frac{\Delta\bar{n}\left(t\right)}{N\Delta t} \tag{5}$$

By employing the mean-field approximation, we can express the current and membrane potential as depicted in Equation 6 and Equation 7, respectively.

$$RI(t)=\tau_{\mathrm{m}}\sum_{\xi=1}^{X}p^{\xi}N^{\xi}w^{\xi}\left(\epsilon^{\xi}*A^{\xi}\right)(t) \tag{6}$$

$$\tau_{\mathrm{m}}\frac{\partial u}{\partial t}=-u+\mu(t)+\tau_{\mathrm{m}}\sum_{\xi=1}^{X}p^{\xi}N^{\xi}w^{\xi}\left(\epsilon^{\xi}*A^{\xi}\right)(t) \tag{7}$$

In the above equations, the subscript and superscript i are omitted.

In the following, the subscript i denotes the individual neuron and the subscript A denote the population. With the quasi-renewal approximation, the population firing rate $\lambda(t)$ is reformulated into a simpler yet effective format, $\lambda_i(t \mid \hat{t}_i)$, which considers only the most recent spike $\hat{t}_i$ of the neuron and the historical population activity. In the simplified $\lambda_i(t)\approx f(u_A(t,\hat{t}_i)-\vartheta_i(t))$, the threshold $\vartheta_i(t)$ is determined by all the spikes in the firing history, which is not renewal-type – it is not merely time-dependent. The quasi-renewal approximation uses an adaptation kernel to describe the effect of spikes preceding the last spike, which can be considered as the "average" of those spikes (the spike activities of neurons correspond to an inhomogeneous Poisson process). Then we have $\lambda_i(t)\approx f(u_A(t,\hat{t}_i)-\vartheta_A(t,\hat{t}_i))$, denoted as $\lambda_A(t \mid \hat{t}_i)$, where $\vartheta_A(t,\hat{t})=u_{\mathrm{th}}+\theta(t-\hat{t})+\int_{-\infty}^{\hat{t}}\tilde{\theta}\left(t-\tau\right)A(\tau)\mathrm{d}\tau$ and $\tilde{\theta}(t)=\Delta_u[1-e^{-\theta(t)/\Delta_u}]$. This is an approximation based on the microscopic model. Δ_u is the softness coefficient. u_{th} is the potential threshold. $\theta(t)$ here is the basic kernel of the adaptation of spikes on the potential threshold, same as that in the microscopic model, which is $\theta(t)=\frac{J_\theta}{\tau_\theta}e^{-t/\tau_\theta}$, where $\frac{1}{\tau_\theta}e^{-t/\tau_\theta}$ is a normalized term with area under the curve being 1, and J_θ and τ_θ are adaptation strength and adaptation time scale, respectively. Note that $\theta(t)$ can include multiple exponential terms, i.e., $\theta(t)=\sum_z\frac{J_{\theta,z}}{\tau_{\theta,z}}e^{-t/\tau_{\theta,z}}$. In our cases, we use the single exponential function.

Let's introduce some essential intermediate terms for mesoscopic description. $m(t_1,t_2)$ is the count of neurons at time t_1 having their last spikes at time t_2, with $\sum_{k=-\infty}^{t_{l-1}}m(t_1,t_2)=N$. Survival rate $S(t_1\mid t_2)$ is defined as the ratio of the mean neuron count $\langle\hat{m}(t_1,t_2)\rangle$ to the change in neuron count $\Delta n(t_2)$. The variance $v\left(t_l,t_k\right)$ is determined as the mean change in squared neuron count $\langle\Delta\hat{m}^2(t_1,t_2)\rangle$ divided by $N\Delta t$. The mean $\langle\hat{m}(t_1,t_2)\rangle$ represents a mesoscopic variable. All these variables require initial conditions to commence the simulation of the mesoscopic model [10, 23–25].

By transitioning from microscopic to mesoscopic variables, we reach a point of low-dimensional variable space without being overly coarse. Finally, after certain statistical derivations such as approximation from binomial distribution to Poisson distribution, from a sufficient number of Poisson distributions to Gaussian distribution and transformation of statistical characteristics, we obtain the following equations:

$$A(t)=\bar{A}(t)+\sqrt{\frac{\bar{A}(t)}{N}}\zeta(t) \tag{8}$$

where $\zeta(t)$ is a Gaussian white noise, and

$$\bar{A}(t) = \int_{-\infty}^{t} \lambda_A(t \mid \hat{t}) S(t \mid \hat{t}) A(\hat{t}) \mathrm{d}\hat{t} + \Lambda(t) \left(1 - \int_{-\infty}^{t} S(t \mid \hat{t}) A(\hat{t}) \mathrm{d}\hat{t} \right) \tag{9}$$

$$\lambda_A(t \mid \hat{t}) = c \exp\left(\frac{u(t,\hat{t}) - \vartheta_A(t,\hat{t})}{\Delta_u} \right), \quad \Lambda(t) = \frac{\int_{-\infty}^{t} \lambda_A(t \mid \hat{t}) v(t,\hat{t}) \mathrm{d}\hat{t}}{\int_{-\infty}^{t} v(t,\hat{t}) \mathrm{d}\hat{t}} \tag{10}$$

B Simulation time comparison

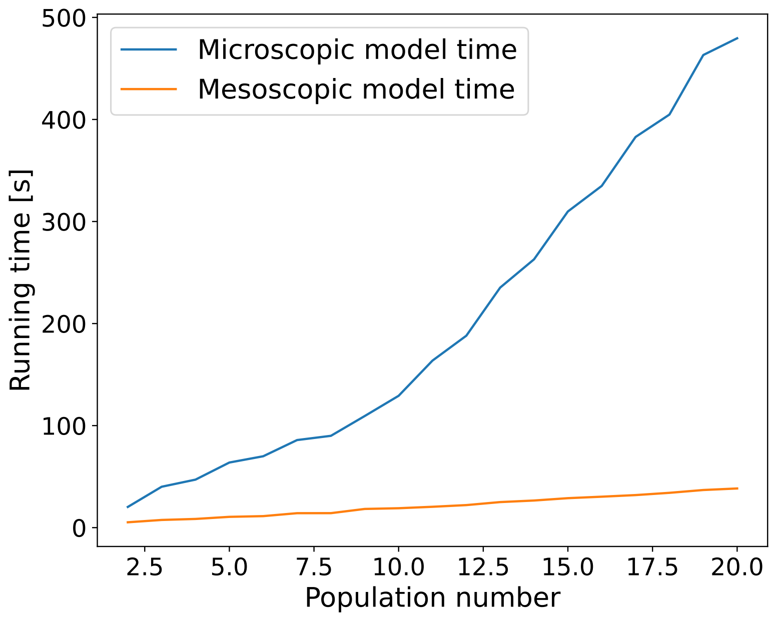

Figure 1: **Running time: mesoscopic vs. microscopic models** As population increases, mesoscopic model demonstrates significantly shorter running time compared to microscopic model, making it the preferred choice for data collection.

Figure 1 illustrates the comparison between the running time of mesoscopic and microscopic models as the population number increases. The graph clearly demonstrates that the mesoscopic model exhibits significantly shorter running time compared to the microscopic model. This finding highlights the advantage of using the mesoscopic model for data collection purposes.

Upon analyzing the data, it is clear that the mesoscopic model vastly outperforms the microscopic model in computational efficiency as the number of neuron populations increases. Specifically, at a population size of 20, the mesoscopic model's running time is approximately 40 seconds, while the microscopic model takes a staggering 480 seconds, more than 12 times longer. However, it is important to note that these timings are based on each population having a consistent size of 200 neurons. The computational performance of the microscopic model is particularly sensitive to the number of neurons, meaning its running time would exponentially increase with a larger neuron count per population.

In the context of extensive data collection—encompassing thousands, if not tens of thousands of records—this difference in computational efficiency becomes a decisive factor. As the computational burden of the microscopic model escalates rapidly with the number of neurons and populations, it becomes increasingly impractical for large-scale data collection (e.g., the network in Table 1).

In contrast, the mesoscopic model's performance scales more efficiently with the increase in neuron populations, making it a far more feasible option for large-scale operations. Thus, the use of the mesoscopic model is not merely preferable, but practically essential for extensive data collection in the realm of spiking neural network parameter estimation. This model's superior computational speed, combined with its ability to maintain a similar degree of accuracy to the microscopic model, positions it as the optimal choice for robust data collection and in-depth analysis, ultimately facilitating a more comprehensive understanding of neural dynamics.

C Parameter settings, data collection, and data processing

Parameter settings In addition to the key parameters that require estimation, including synaptic weights between each pair of populations (J_{syn}), resting potential (μ, which combines the resting potential and the constant external input due to their similar effects on neuronal activities), membrane time constant (τ_m), baseline threshold voltage (u_{th}), adaptation strength (J_θ), and adaptation time scale (τ_θ), we hold other parameters constant. In our case studies, we consider two scenarios: networks with 3 populations and 8 populations. The network architecture for both the 3-population and 8-population scenarios are depicted in Figure 2.

In the 3-population scenario, the number of neurons in each of the three populations is 400, 200, and 400 respectively. The base rate for the exponential link function is set to 10 Hz, the reset potential is at 0 mV, the refractory period is 4 ms, the threshold softness for adaptation is 2.5 mV, the transmission delay constant is 1 ms, and the time constants for excitatory and inhibitory synapses are 3 ms and 6 ms, respectively. The connection probabilities are all set to 0.6. If the biological simulation time is 60 s, we have a step input of 20 mV starting at the 30 s mark; otherwise, there is no step input.

For the 8-population scenario, which applies to a modified Potjans-Diesmann model network, parameter settings can be found in Table 1 and Table 2 [10, 26]. Any parameters not listed in these two tables are consistent with those set in the 3-population scenario.

Data collection For the key parameters that need to be estimated – including J_{syn}, μ, τ_m, u_{th}, J_θ, and τ_θ – we generate data randomly within a reasonable range. Owing to the randomness of J_{syn}, it is not certain whether each population is excitatory or inhibitory.

Data processing Let's denote the total number of populations as Y and the number of exponential kernels used to approximate the adaptation from the spikes before the last spike as Z. As such, the six parameters – the synaptic connectivity matrix J_{syn}, resting potential μ, membrane time constant τ_m, baseline threshold voltage u_{th}, adaptation strength J_θ, and adaptation time scale τ_θ – have dimensions of $Y \times Y$, Y, Y, Y, $Y \times Z$, and $Y \times Z$, respectively.

Before feeding data into the RNN, we utilize an averaging operation to down-sample the data into shorter sequences for speedier training and inference. During data normalization, after collecting a certain amount of data, we find the maximum absolute value among this data set and divide all data points by this maximum. This results in normalized values between -1 and 1 (or 0 and 1 if all values are positive). Notably, for the J_{syn} data, the maximum value is calculated independently for each population. To revert to the original data from the output, we must denormalize the output using the previously determined normalization factors.

D Training and loss

Our recurrent neural network (RNN), equipped with the gated recurrent unit (GRU), consists of one to three stacked recurrent layers and two or three fully connected layers connected to the GRU output. The mean squared error (MSE) loss, based on normalized input, is utilized.

The training and validation losses for both the 3-population and 8-population scenarios are illustrated in Figure 3. The loss curves indicate satisfactory training for both scenarios. However, the RNN model with GRU demonstrates superior performance in the parameter estimation for the 3-population scenario. This is likely due to the higher complexity of the 8-population scenario, exemplified by the increased dimensions of the 8x8 J_{syn} matrix, which results in greater losses. To enhance training performance for the 8-population scenario, we might contemplate utilizing a weighted MSE loss, assigning greater weights to more challenging regressions, such as J_{syn}. In essence, as the number of populations escalates, further training techniques warrant exploration.

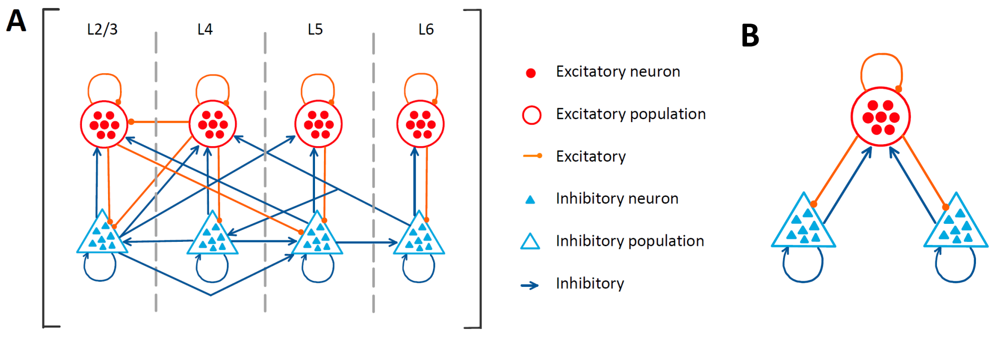

Figure 2: **Schematic of network architecture.** **(A)** The architecture of the eight-population is composed of four layers (L2/3, L4, L5, and L6). Each layer comprises both excitatory and inhibitory populations. **(B)** The architecture of the three-population comprises two competing excitatory populations and a shared inhibitory population, representing a winner-take-all SNN.

Table 1: Parameters of the modified Potjans-Diesmann model.

population	**L2/3e**	**L2/3i**	**L4e**	**L4i**	**L5e**	**L5i**	**L6e**	**L6i**
synaptic time constants [s]	0.0005							
transmission constant delay [s]	0.0015							
refractory period [s]	0.002							
softness of threshold adaptation [mV]	5.0							
external step input time	[0.06s, 0.09s]							
step stimulus [mV]	0	0	19	11.964	0	0	9.896	3.788
neuron number	20683	5834	21915	5479	4850	1065	14395	2948

E Comparison of groundtruth and estimation

Table 3, Table 4, Table 5, Table 6, Table 7, and Figure 4, Figure 5, Figure 6, Figure 7, Figure 8 show the numerical and graphical results of 5 samples under the 3-population scenario. Table 8, Table 9, Table 10, Table 11, Table 12, Table 13, Table 14, Table 15, and Figure 9 and Figure 10 show the numerical and graphical results of 2 samples under the 8-population scenario. To enhance clarity, we show curves of population activities that have undergone low-pass filtering.

Considering that minor variations in J_{syn} can lead to significant differences in neuronal activities, we utilize the J_{syn} from the label and other parameters from estimation when plotting neuronal activities based on estimation. However, this does not render the prediction of J_{syn} irrelevant. We can still employ the columns of J_{syn} to discern whether the populations are excitatory or inhibitory. Moreover, we could establish a threshold for elements in J_{syn}; if an element in J_{syn} exceeds the threshold, it signifies a connection between the corresponding two populations, providing a raw connectivity matrix composed of 0s and 1s.

In general, most numerical and graphical results illustrated in the preceding tables and figures exhibit good performance. For instance, in the 3-population scenario, samples 2 and 3 possess the same labels, and despite the distinct neuronal activity inputs due to neuronal stochastic processes, the predictions produce similar outcomes. For sample 5 under the same scenario, we observe a bistability phenomenon typical in a winner-take-all SNN.

Nevertheless, we must acknowledge that the exact estimation of J_{syn} remains challenging. As seen in Table 11 and Table 15, the MSE loss excluding J_{syn} is smaller, suggesting that the major difficulty in

Table 2: Connection probability for 8-population scenario.

	L2/3e	**L2/3i**	**L4e**	**L4i**	**L5e**	**L5i**	**L6e**	**L6i**
L2/3e	0.1009	0.1689	0.0437	0.0818	0.0323	0	0.0076	0
L2/3i	0.1346	0.1371	0.0316	0.0515	0.0755	0	0.0042	0
L4e	0.0077	0.0059	0.0497	0.135	0.0067	0.0003	0.0453	0
L4i	0.0691	0.0029	0.0794	0.1597	0.0033	0	0.1057	0
L5e	0.1004	0.0622	0.0505	0.0057	0.0831	0.3726	0.0204	0
L5i	0.0548	0.0269	0.0257	0.0022	0.06	0.3158	0.0086	0
L6e	0.0156	0.0066	0.0211	0.0166	0.0572	0.0197	0.0396	0.2252
L6i	0.0364	0.001	0.0034	0.0005	0.0277	0.008	0.0658	0.1443

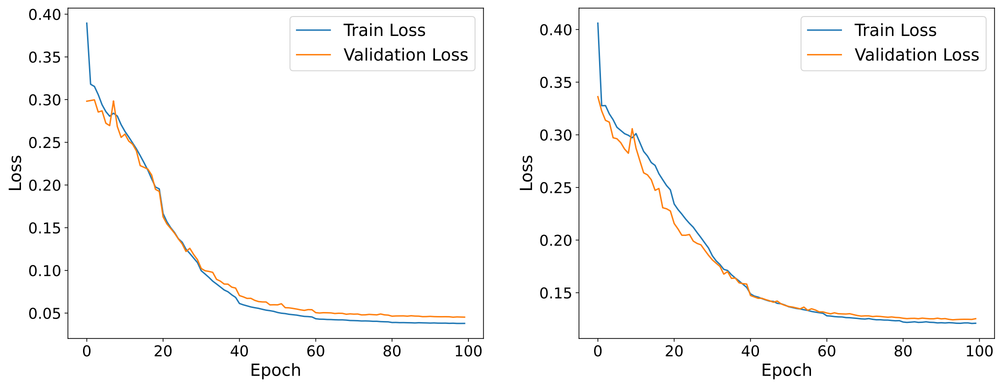

Figure 3: **Training loss and validation loss.** The left and right figures display the loss in 3-population case and 8-population cases, respectively.

estimation lies with J_{syn}. Further theoretical insights and engineering techniques need exploration to either enhance the estimation of J_{syn} or to prove its inherent complexity. Apart from J_{syn}, the model performs well for other parameters.

F Discussion of RNNs versus CNNs

Neurological sequence data typically exhibit temporal dependencies, where the state at a given time is influenced by the states at preceding times. Why we use RNNs not convolutional neural networks (CNNs) for handling this type of data? (1) Temporal Dynamics: RNNs are fundamentally designed to handle sequential data. They take into account the temporal dynamics of data, as they possess "memory" in the form of hidden states. CNNs, even though they have proven effective in capturing spatial hierarchies in data, do not inherently account for temporal dependencies (2) Variable Length Input: RNNs can handle variable-length inputs, which is particularly beneficial when dealing with neurological sequences. CNNs, in contrast, typically require fixed-size inputs. (3) Continuous Time Information: RNNs can process sequences with time steps that are spaced unevenly or continuously, which can be quite relevant for neurological sequence data. CNNs typically assume evenly spaced data points, which may not always be the case in real-world neurological sequence data.

Moreover, the attention mechanism has a more natural fit within the sequential processing paradigm of RNNs. This compatibility potentially paves the way for enhanced performance in our future work. However, CNNs can still have utility as a preprocessing step before RNNs for sequence data, such as feature extraction or/and dimensionality reduction. The combined use of CNNs and RNNs can provide a robust method for analyzing neurological sequence data, leveraging the strengths of both architectures, which might also be our future work.

Table 3: Numerical comparison of groundtruth and estimation (3-population, sample 1).

		Labels			**Estimations**		
		P1	P2	P3	P1	P2	P3
J_{syn} **[mV]**	P1	-0.56	-0.27	0.00	-0.76	-0.13	0.20
	P2	-0.58	-0.27	0.07	-0.42	-0.30	-0.09
	P3	-0.60	-0.27	0.12	-0.58	-0.21	0.20
μ **[mV]**		48.94	50.31	30.73	49.66	49.87	30.55
τ_m **[ms]**		27.12	19.11	35.79	27.14	20.30	35.60
u_{th} **[mV]**		28.26	23.84	12.17	28.03	24.84	11.85
J_θ **[mV·ms]**		971.74	971.74	144.08	977.28	986.23	185.08
τ_θ **[ms]**		804.97	804.97	1340.50	799.58	790.15	1345.05
Normalized MSE loss				0.0053			

Table 4: Numerical comparison of groundtruth and estimation (3-population, sample 2).

		Labels			**Estimations**		
		P1	P2	P3	P1	P2	P3
J_{syn} **[mV]**	P1	-0.32	0.00	0.07	-0.46	-0.46	-0.06
	P2	-0.29	-0.32	0.00	-0.31	-0.43	-0.01
	P3	0.00	0.00	0.10	-0.03	-0.09	-0.14
μ **[mV]**		34.07	55.31	37.28	33.62	53.50	38.11
τ_m **[ms]**		27.27	18.28	20.93	27.75	18.82	21.21
u_{th} **[mV]**		11.42	16.40	26.81	12.17	16.43	26.87
J_θ **[mV·ms]**		674.67	674.67	1231.18	665.73	691.55	1246.66
τ_θ **[ms]**		508.75	508.75	879.77	496.08	498.71	888.23
Normalized MSE loss				0.0040			

Table 5: Numerical comparison of groundtruth and estimation (3-population, sample 3).

		Labels			**Estimations**		
		P1	P2	P3	P1	P2	P3
J_{syn} **[mV]**	P1	-0.32	0.00	0.07	-0.46	-0.11	-0.06
	P2	-0.29	-0.32	0.00	-0.33	-0.44	-0.01
	P3	0.00	0.00	0.10	-0.02	-0.10	-0.15
μ **[mV]**		34.07	55.31	37.28	33.42	54.23	38.04
τ_m **[ms]**		27.27	18.28	20.93	27.96	18.40	21.03
u_{th} **[mV]**		11.42	16.40	26.81	12.09	16.17	26.58
J_θ **[mV·ms]**		674.67	674.67	1231.18	685.28	700.26	1230.59
τ_θ **[ms]**		508.75	508.75	879.77	513.01	509.85	904.13
Normalized MSE loss				0.0041			

Table 6: Numerical comparison of groundtruth and estimation (3-population, sample 4).

		Labels			**Estimations**		
		P1	P2	P3	P1	P2	P3
J_{syn} **[mV]**	P1	0.09	0.00	0.00	0.10	0.07	0.07
	P2	0.00	0.07	0.00	0.13	0.00	0.09
	P3	0.00	0.06	0.08	-0.11	0.01	0.16
μ **[mV]**		28.58	31.18	29.20	26.61	31.75	28.94
τ_m **[ms]**		36.08	33.52	27.83	37.19	33.08	28.09
u_{th} **[mV]**		26.91	23.43	27.28	26.22	24.17	26.87
J_θ **[mV·ms]**		256.46	256.46	256.46	284.75	280.75	287.15
τ_θ **[ms]**		676.02	676.02	676.02	685.15	621.94	644.17
Normalized MSE loss		0.0036					

Table 7: Numerical comparison of groundtruth and estimation (3-population, sample 5).

		Labels			**Estimations**		
		P1	P2	P3	P1	P2	P3
J_{syn} **[mV]**	P1	0.16	-0.64	0.00	0.16	-0.70	-0.10
	P2	0.16	-0.64	0.16	0.18	-0.62	0.17
	P3	0.00	-0.64	0.16	0.05	-0.67	0.19
μ **[mV]**		36.00	36.00	36.00	35.04	35.98	35.41
τ_m **[ms]**		20.00	20.00	20.00	20.62	20.64	20.84
u_{th} **[mV]**		15.00	15.00	15.00	14.98	15.03	14.97
J_θ **[mV·ms]**		100.00	100.00	100.00	112.60	96.41	108.94
τ_θ **[ms]**		1000.00	1000.00	1000.00	1074.90	1037.52	1144.01
Normalized MSE loss		0.0051					

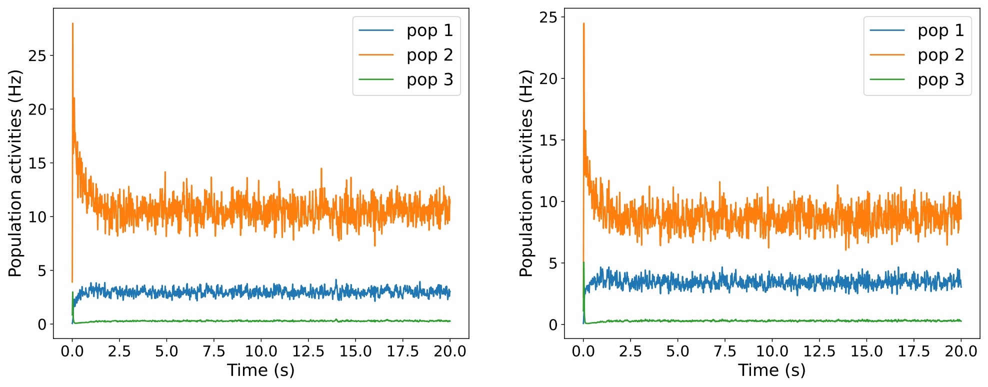

Figure 4: **Neuron activities comparison (3-population, sample 1).** The left figure displays the label result, and the right figure displays the estimation result.

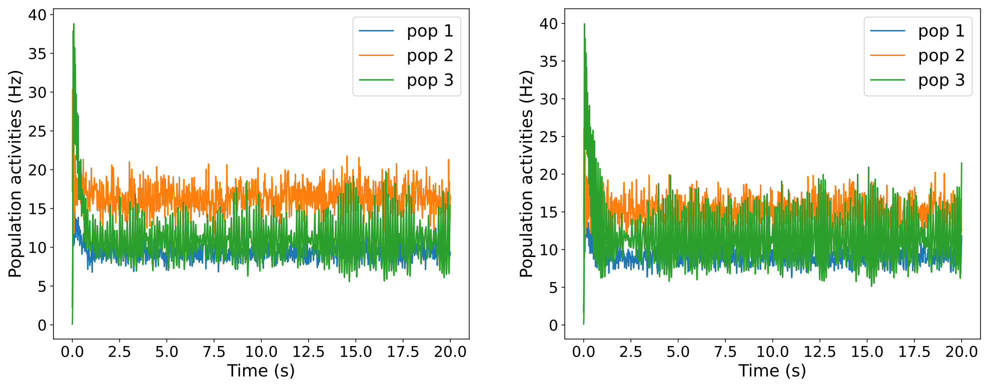

Figure 5: **Neuron activities comparison (3-population, sample 2).** The left figure displays the label result, and the right figure displays the estimation result.

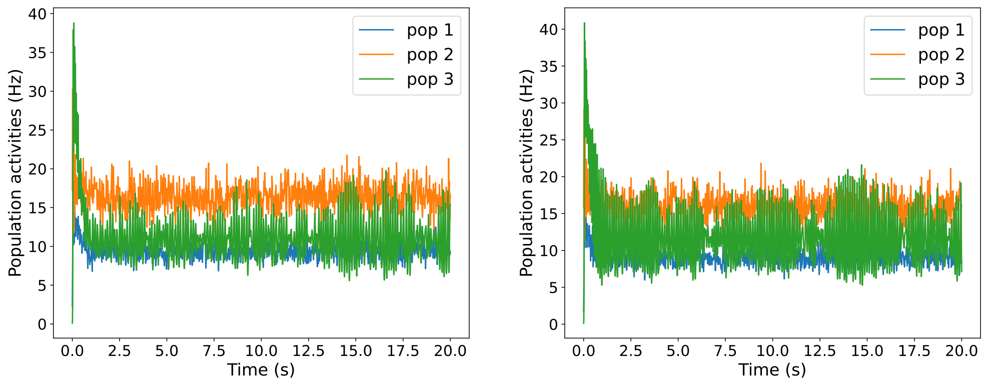

Figure 6: **Neuron activities comparison (3-population, sample 3).** The left figure displays the label result, and the right figure displays the estimation result.

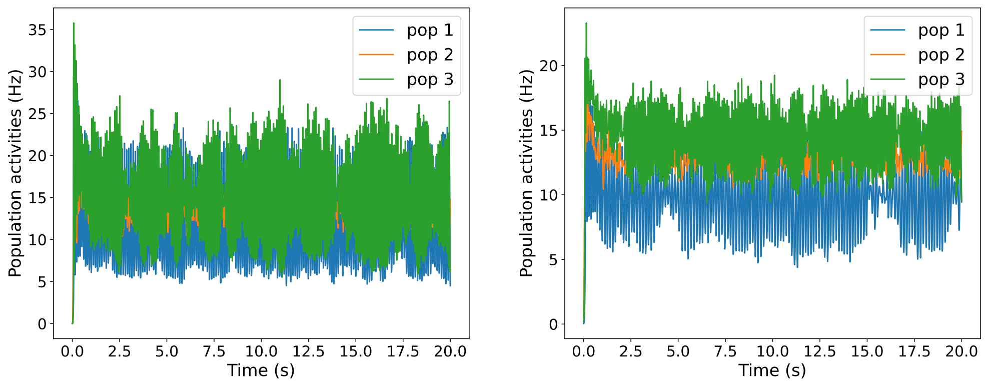

Figure 7: **Neuron activities comparison (3-population, sample 4).** The left figure displays the label result, and the right figure displays the estimation result.

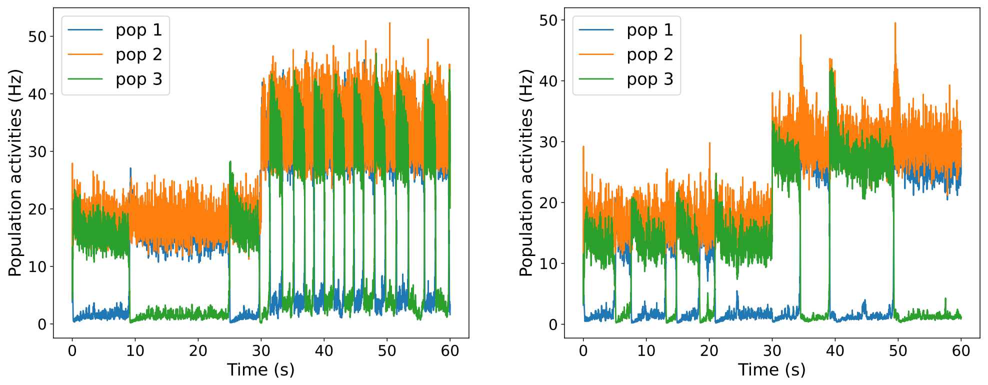

Figure 8: **Neuron activities comparison (3-population, sample 5).** The left figure displays the label result, and the right figure displays the estimation result.

Table 8: Numerical comparison of groundtruth and estimation (8-population, sample 1 part 1: the labels of J_{syn}).

		Labels							
		P1	P2	P3	P4	P5	P6	P7	P8
J_{syn} **[mV]**	P1	-0.66	0.20	0.18	0.00	0.00	-0.58	0.00	0.00
	P2	-0.40	0.15	0.00	0.00	0.17	0.00	0.13	0.00
	P3	-0.43	0.17	0.18	-0.41	0.14	-0.71	0.00	0.00
	P4	-0.43	0.22	0.00	-0.87	0.00	0.00	0.00	0.21
	P5	-0.92	0.00	0.19	0.00	0.17	-1.03	0.00	0.09
	P6	-0.49	0.00	0.12	-1.11	0.00	-0.90	0.12	0.16
	P7	-0.95	0.00	0.00	-0.75	0.15	-0.40	0.11	0.22
	P8	0.00	0.00	0.00	0.00	0.13	-0.57	0.00	0.11

Table 9: Numerical comparison of groundtruth and estimation (8-population, sample 1 part 2: the estimations of J_{syn}).

		Estimations							
		P1	P2	P3	P4	P5	P6	P7	P8
J_{syn} **[mV]**	P1	-0.84	0.14	-0.08	-0.51	0.29	-0.45	0.01	0.21
	P2	-0.52	-0.25	-0.14	-0.48	-0.04	-0.03	0.01	0.57
	P3	-0.36	-0.05	0.03	-0.51	-0.16	-0.79	-0.08	0.03
	P4	-0.34	0.19	-0.13	-1.26	-0.21	-0.45	-0.17	0.04
	P5	-1.03	-0.39	-0.07	-0.48	0.03	-0.66	-0.12	0.30
	P6	-0.40	-0.43	0.09	-1.40	0.05	-0.92	0.10	-0.06
	P7	-1.02	-0.31	0.01	-0.82	0.19	-0.20	-0.02	0.46
	P8	-0.23	-0.04	0.04	-0.14	0.20	-0.31	0.05	0.38

Table 10: Numerical comparison of groundtruth and estimation (8-population, sample 1 part 3: the labels of μ, τ_m, u_{th}, J_θ, τ_θ).

	Labels							
	P1	P2	P3	P4	P5	P6	P7	P8
μ **[mV]**	51.52	54.30	26.38	20.55	32.86	53.42	22.34	42.00
τ_m **[ms]**	30.92	19.33	15.19	28.60	23.90	31.52	23.31	16.49
u_{th} **[mV]**	13.61	24.40	28.83	12.28	19.78	10.91	24.40	18.69
J_θ **[mV·ms]**	982.48	43.80	43.80	982.48	43.80	982.48	43.80	43.80
τ_θ **[ms]**	1080.22	577.58	577.58	1080.22	577.58	1080.22	577.58	577.58

Table 11: Numerical comparison of groundtruth and estimation (8-population, sample 1 part 4: the estimations of μ, τ_m, u_{th}, J_θ, τ_θ and normalized MSE loss with/without J_{syn}).

	Estimations							
	P1	P2	P3	P4	P5	P6	P7	P8
μ **[mV]**	51.41	50.04	33.05	23.89	28.71	56.63	19.23	42.77
τ_m **[ms]**	30.87	15.79	20.34	31.98	25.12	31.58	19.62	20.35
u_{th} **[mV]**	12.61	25.36	29.32	13.90	19.09	11.99	21.24	23.26
J_θ **[mV·ms]**	872.58	180.64	70.87	1204.65	133.53	1017.42	22.94	87.04
τ_θ **[ms]**	998.88	539.75	485.23	1209.86	723.77	1187.07	785.79	828.82
Normalized MSE loss	0.0440							
Normalized MSE loss without J_{syn}	0.0303							

Table 12: Numerical comparison of groundtruth and estimation (8-population, sample 2 part 1: the labels of J_{syn}).

		Labels							
		P1	P2	P3	P4	P5	P6	P7	P8
J_{syn} **[mV]**	P1	0.05	0.06	0.06	0.00	0.00	-0.32	0.00	0.11
	P2	0.00	0.11	0.12	0.00	0.00	0.00	0.09	0.07
	P3	0.00	0.08	0.08	-0.35	0.11	0.00	0.06	0.10
	P4	0.06	0.08	0.00	-0.45	0.09	0.00	0.00	0.00
	P5	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00
	P6	0.12	0.00	0.00	-0.46	0.00	-0.46	0.09	0.00
	P7	0.07	0.00	0.06	-0.19	0.12	-0.37	0.12	0.04
	P8	0.11	0.08	0.10	0.00	0.06	0.00	0.00	0.07

Table 13: Numerical comparison of groundtruth and estimation (8-population, sample 2 part 2: the estimations of J_{syn}).

		Estimations							
		P1	P2	P3	P4	P5	P6	P7	P8
J_{syn} **[mV]**	P1	-0.30	0.14	-0.05	-0.23	-0.05	-0.54	-0.13	-0.11
	P2	0.10	-0.12	-0.16	-0.10	-0.06	-0.63	-0.04	0.22
	P3	0.07	0.13	-0.29	-0.15	0.19	-0.16	-0.18	-0.15
	P4	-0.15	0.09	0.03	-0.46	0.02	-0.39	0.03	0.12
	P5	-0.03	0.06	-0.01	-0.16	-0.30	-0.23	0.08	0.03
	P6	-0.06	-0.14	-0.07	-0.18	0.01	-0.58	0.05	0.28
	P7	0.02	-0.22	-0.28	-0.14	0.02	-0.16	-0.15	-0.01
	P8	-0.09	-0.13	-0.04	-0.31	0.06	-0.32	-0.01	0.10

Table 14: Numerical comparison of groundtruth and estimation (8-population, sample 2 part 3: the labels of μ, τ_m, u_{th}, J_θ, τ_θ).

	Labels							
	P1	P2	P3	P4	P5	P6	P7	P8
μ **[mV]**	43.73	48.50	49.12	46.45	57.00	55.25	55.04	56.00
τ_m **[ms]**	35.51	17.44	24.96	29.30	26.67	11.86	24.74	27.86
u_{th} **[mV]**	19.90	12.80	24.18	26.43	21.54	18.83	20.23	18.44
J_θ **[mV·ms]**	1152.54	1152.54	1152.54	99.35	1152.54	99.35	1152.54	1152.54
τ_θ **[ms]**	1146.80	1146.80	1146.80	651.40	1146.80	651.40	1146.80	1146.80

Table 15: Numerical comparison of groundtruth and estimation (8-population, sample 2 part 4: the estimations of μ, τ_m, u_{th}, J_θ, τ_θ and normalized MSE loss with/without J_{syn}).

	Estimations							
	P1	P2	P3	P4	P5	P6	P7	P8
μ **[mV]**	49.85	47.78	48.46	42.75	48.12	55.19	54.77	58.31
τ_m **[ms]**	39.89	24.80	25.22	32.39	27.76	12.87	26.32	29.04
u_{th} **[mV]**	20.78	10.85	21.97	23.94	19.03	18.65	16.40	16.98
J_θ **[mV·ms]**	970.41	1270.93	764.44	473.48	1075.95	317.52	1180.89	1042.70
τ_θ **[ms]**	1311.60	1301.01	1140.90	845.82	926.21	815.45	1348.87	1116.72
Normalized MSE loss				0.0617				
Normalized MSE loss without J_{syn}				0.0518				

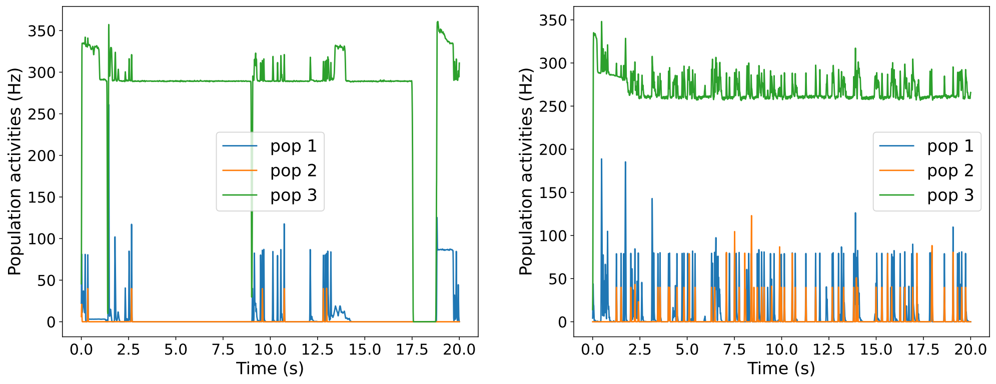

Figure 9: **Neuron activities comparison (8-population, sample 1).** 3 populations are randomly selected from 8 populations for display. The left figure displays the label result, and the right figure displays the estimation result.

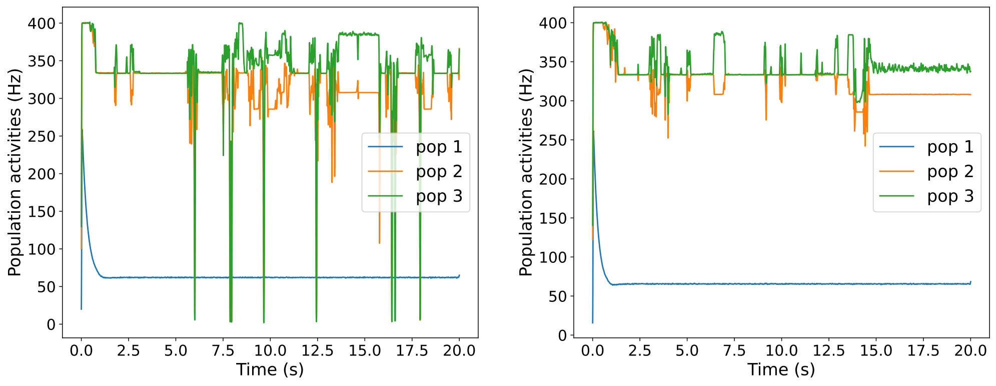

Figure 10: **Neuron activities comparison (8-population, sample 2).** 3 populations are randomly selected from 8 populations for display. The left figure displays the label result, and the right figure displays the estimation result.

References

[1] Sandrine Lefort, Christian Tomm, J-C Floyd Sarria, and Carl CH Petersen. The excitatory neuronal network of the c2 barrel column in mouse primary somatosensory cortex. *Neuron*, 61(2):301–316, 2009.

[2] Christian Pozzorini, Skander Mensi, Olivier Hagens, Richard Naud, Christof Koch, and Wulfram Gerstner. Automated high-throughput characterization of single neurons by means of simplified spiking models. *PLoS computational biology*, 11(6):e1004275, 2015.

[3] Scott J Cruikshank, Timothy J Lewis, and Barry W Connors. Synaptic basis for intense thalamocortical activation of feedforward inhibitory cells in neocortex. *Nature neuroscience*, 10(4):462–468, 2007.

[4] Mahesh M Karnani, Jesse Jackson, Inbal Ayzenshtat, Jason Tucciarone, Kasra Manoocheri, William G Snider, and Rafael Yuste. Cooperative subnetworks of molecularly similar interneurons in mouse neocortex. *Neuron*, 90(1):86–100, 2016.

[5] Anita Tusche, Anne Böckler, Philipp Kanske, Fynn-Mathis Trautwein, and Tania Singer. Decoding the charitable brain: empathy, perspective taking, and attention shifts differentially predict altruistic giving. *Journal of Neuroscience*, 36(17):4719–4732, 2016.

[6] Carsten K Pfeffer, Mingshan Xue, Miao He, Z Josh Huang, and Massimo Scanziani. Inhibition of inhibition in visual cortex: the logic of connections between molecularly distinct interneurons. *Nature neuroscience*, 16(8):1068–1076, 2013.

[7] Adam M Packer and Rafael Yuste. Dense, unspecific connectivity of neocortical parvalbumin-positive interneurons: a canonical microcircuit for inhibition? *Journal of Neuroscience*, 31(37):13260–13271, 2011.

[8] Luc J Gentet, Yves Kremer, Hiroki Taniguchi, Z Josh Huang, Jochen F Staiger, and Carl CH Petersen. Unique functional properties of somatostatin-expressing gabaergic neurons in mouse barrel cortex. *Nature neuroscience*, 15(4):607–612, 2012.

[9] Michael Avermann, Christian Tomm, Celine Mateo, Wulfram Gerstner, and Carl CH Petersen. Microcircuits of excitatory and inhibitory neurons in layer 2/3 of mouse barrel cortex. *Journal of neurophysiology*, 107 (11):3116–3134, 2012.

[10] Tilo Schwalger, Moritz Deger, and Wulfram Gerstner. Towards a theory of cortical columns: From spiking neurons to interacting neural populations of finite size. *PLoS computational biology*, 13(4):e1005507, 2017.

[11] Shuqi Wang, Valentin Schmutz, Guillaume Bellec, and Wulfram Gerstner. Mesoscopic modeling of hidden spiking neurons. *arXiv preprint arXiv:2205.13493*, 2022.

[12] Y Liu. H, wang xj. *Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. J Comput Neurosci*, 10(1):25–45, 2001.

[13] Maurice J Chacron, André Longtin, Martin St-Hilaire, and Len Maler. Suprathreshold stochastic firing dynamics with memory in p-type electroreceptors. *Physical Review Letters*, 85(7):1576, 2000.

[14] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[15] C Daniel Geisler and Jay M Goldberg. A stochastic model of the repetitive activity of neurons. *Biophysical journal*, 6(1):53–69, 1966.

[16] Christian Pozzorini, Richard Naud, Skander Mensi, and Wulfram Gerstner. Temporal whitening by power-law adaptation in neocortical neurons. *Nature neuroscience*, 16(7):942–948, 2013.

[17] Skander Mensi, Richard Naud, Christian Pozzorini, Michael Avermann, Carl CH Petersen, and Wulfram Gerstner. Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms. *Journal of neurophysiology*, 107(6):1756–1775, 2012.

[18] Moritz Deger, Tilo Schwalger, Richard Naud, and Wulfram Gerstner. Fluctuations and information filtering in coupled populations of spiking neurons with adaptation. *Physical Review E*, 90(6):062704, 2014.

[19] Taro Toyoizumi, Kamiar Rahnama Rad, and Liam Paninski. Mean-field approximations for coupled populations of generalized linear model spiking neurons with markov refractoriness. *Neural computation*, 21(5):1203–1243, 2009.

[20] Alison I Weber and Jonathan W Pillow. Capturing the dynamical repertoire of single neurons with generalized linear models. *Neural computation*, 29(12):3260–3289, 2017.

[21] Wilson Truccolo, Uri T Eden, Matthew R Fellows, John P Donoghue, and Emery N Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of neurophysiology*, 93(2):1074–1089, 2005.

[22] Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

[23] Valentin Schmutz, Eva Löcherbach, and Tilo Schwalger. On a finite-size neuronal population equation. *arXiv preprint arXiv:2106.14721*, 2021.

[24] Tilo Schwalger and Anton V Chizhov. Mind the last spike—firing rate models for mesoscopic populations of spiking neurons. *Current opinion in neurobiology*, 58:155–166, 2019.

[25] Alexandre René, André Longtin, and Jakob H Macke. Inference of a mesoscopic population model from population spike trains. *Neural computation*, 32(8):1448–1498, 2020.

[26] C Tobias. Potjans and markus diesmann. the cell-type specific cortical microcircuit: Relating structure and activity in a full-scale spiking network model. *Cerebral Cortex*, (24):785, 2014.