Metamodelling of a two-population spiking neural network

Jan-Eirik W. Skaar[1], Alexander J. Stasik[3], Hans Ekkehard Plesser[1], Gaute T. Einevoll[1,2], Kristin Tøndel[1]

1 Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, Norway

2 Department of Physics, University of Oslo, Oslo, Norway

3 SINTEF, Department of Mathematics and Cybernetics, Oslo, Norway

* Corresponding author: jan-eirik.welle.skaar@nmbu.no

Abstract

In computational neuroscience, hypotheses are often formulated as bottom-up mechanistic models of the systems in question, consisting of differential equations that can be numerically integrated forward in time. Candidate models can then be validated by comparison against experimental data. The model outputs of neural network models depend on both neuron parameters, connectivity parameters and other model inputs. Successful model fitting requires sufficient exploration of the model parameter space, which can be computationally demanding. Additionally, identifying degeneracy in the parameters, i.e. different combinations of parameter values that produce similar outputs, is of interest, as they define the subset of models consistent with the data. In this computational study, we apply metamodels to a two-population recurrent spiking network of point-neurons, the so-called Brunel network. Metamodels are data-driven approximations to more complex models with more desirable computational properties, which can be run considerably faster than the original model. Specifically, we apply two different metamodelling techniques, Hierarchical Cluster-based Partial Least Squares regression and deep Gaussian process regression, to estimate the power spectra of the population spiking activities and the local field potential. We find that the metamodels are able to accurately model the power spectra in the asynchronous irregular regime. Using the Gaussian process metamodel, we invert the model, and estimate the probability distributions over parameters that can give rise to observed simulation outputs. We find that these distributions correctly identify parameter combinations that give similar model outputs, and that the parameters are constrained to different degrees by observing the two different signals.

Author summary

In computational neuroscience, mechanistic models are used to simulate networks of neurons. These models exhibit complex dynamics, and the parameters of the neurons and connections between neurons shape the model's behaviour. Due to the model complexity, running the simulations and fitting the model to experimental data can be computationally demanding. In this study, we apply metamodelling techniques,

data-driven approximations that are much faster to run, to two different signals generated by a two-population recurrent network model, the population spiking activities and the local field potential (LFP). Further, we invert one of the metamodels, and demonstrate that it can reliably find the different combinations of parameters that can give rise to an observed simulation output, and investigate to what degree the parameters are constrained by the two different signals.

1 Introduction

Mechanistic modelling of neurons and networks of neurons based on the underlying biophysical principles is a well-established field and is an important part in bridging the scales between individual neurons and higher-level brain function [1–4]. Neuron models range in complexity from rate-based models, where spikes are not explicitly modelled, to point-neurons, where the detailed morphology of the individual neurons are collapsed into a single point, and further to multi-compartment models where the full morphology is accounted for. At the network level, even simple models exhibit complex dynamic behaviours depending on both neuron dynamics and network connectivity [5]. A question modellers face is how the network model can be parameterized in order to produce a specific behaviour, and how sensitive the behaviour is to changes in model parameter values. As mechanistic models can be expensive to run and typically have many parameters, that is, high-dimensional parameter spaces, accurately sampling the space of model behaviours is a nontrivial problem.

Mechanistic models of dynamical systems typically consist of a set of differential equations that can be integrated forward in time. The modeller is usually interested in a subset of the model's behaviours, e.g. the state variables of the model, such as membrane potentials, synaptic conductances or currents, individual neuron behaviours such as spike times, or aggregated statistics such as population firing rates, local field potentials [6], or current source densities. The user specifies a set of parameters, which will determine the model's behaviour. If the model is deterministic (no random variables are involved), the output will be a unique function of its parameters and initial conditions. If the model is probabilistic (one or more random variables affect the model), the output will no longer be a unique function of the parameters and initial

conditions, but will instead follow a probability distribution over possible outputs.

Conceptually, a mechanistic model can be thought of as a function $\mathcal{M}$ that takes input values and returns output values,

$$\text{output} = \mathcal{M}(\text{input}) \ .$$

If the model is stochastic, there is a distribution over possible outputs, which typically does not have a closed-form expression, since explicitly evaluating it would require integrating over all possible internal model states. *Metamodels* are data-driven statistical models that take a set of inputs and outputs generated by a mechanistic model, and learn an approximation of $\mathcal{M}$ over a restricted part of its domain [7]. The metamodel can provide either a point-estimate of the model output, or a distribution over possible outputs. In the case of stochastic models, the latter case has the advantage of providing an approximation of the intractable probability distribution over simulator outputs. The number of samples needed to achieve sufficient accuracy over the domain of interest will depend on the dimensionality of the inputs, on how the function $\mathcal{M}$ changes with respect to the input dimensions, and on whether or not it has a structure that the metamodel is able to capture.

The inputs can in principle be anything that changes the model behaviour, e.g. model parameters or some other specific model input. As the volume of the parameter space increases exponentially with the number of parameters, densely sampling high-dimensional parameter spaces is very costly. Metamodels can provide a compact representation of the model over the domain, where the density of sampling required can be determined by evaluating the model on a test data set. Metamodels can also provide simpler interpretations of the model, identify the importance of individual input dimensions, and give an overview of the original model's behavioral repertoire, i.e. a statistical depiction of the possible outputs the model can produce under a range of input conditions [8]. Metamodels can be applied in either direction, from model input to model output, or vice versa, directly approximating the inverse model.

For stochastic models, the distribution over model outputs will depend on the parameter values and possibly other model inputs, giving a conditional probability $p(\mathbf{x}|\boldsymbol{\theta})$, often called the *model likelihood*, where $\mathbf{x}$ is the model output and $\boldsymbol{\theta}$ represents

model parameters and other types of inputs. Since different model inputs possibly can give rise to the same model outputs, there is also a distribution over model inputs that can give rise to the same model outputs, $p(\boldsymbol{\theta}|\mathbf{x})$, often called the *posterior distribution*. Estimating this posterior distribution by iteratively running simulations without direct access to the model likelihood has been the focus of work such as Approximate Bayesian computation (ABC) [9, 10], Sequential Neural Posterior Estimation (SNPE) [11, 12] and Sequential Neural Likelihood Estimation (SNLE) [13]. These methods are primarily focused on estimating the posterior probability $p(\boldsymbol{\theta}|\mathbf{x}_0)$ for a specific observed output $\mathbf{x}_0$, and generating a local model of the posterior distribution around that point by iteratively running simulations where the posterior density is believed to be high. SNPE models the posterior directly, while SNLE models the likelihood, and infers the posterior in a separate step. The latter approach can be viewed as a metamodel that is trained locally around a specific point of interest in the parameter space. Since metamodels can represent the model likelihood, they can also be used in the estimation of posterior distributions through standard methods such as Markov Chain Monte Carlo (MCMC).

Neuronal networks typically exhibit stochastic dynamics, where input currents and firing may be stochastic, and the network connectivity and initial conditions can be drawn randomly. The network activity is also nonlinearly dependent on the precise history of the activity [14]. This makes direct metamodelling of time series data challenging, as different instantiations of the same model will produce different results, which the metamodel will have to account for. Simplified mechanistic models of population rates have been developed, ranging from models based on mean-field theory [5, 14] where the number of neurons is assumed to be infinite and finite-size effects are disregarded, to ones where finite-size effects are fully accounted for [15], but where stochastic firing of neurons and full connectivity between all neurons are assumed.

In the present work, we train metamodels to represent the power spectra of the population spiking activities and local field potential (LFP), based on the model parameters of the Brunel network [5], comprising an excitatory and an inhibitory population of recurrently connected point-neurons. The local field potential, the low-frequency part of extracellular potentials, largely represents how synaptic inputs are processed in the dendrites [6, 16, 17], so we chose to include both signals as they in a sense represent the incoming and outgoing signals for a population, and it offers the

opportunity to compare the information carried by the two different signals.

We train and compare metamodels based on Gaussian process regression (GPR) and Hierarchical Cluster-based Partial Least Squares Regression (HCPLSR) [18], and show that the high-dimensional power spectra of the population spiking activities and LFPs can be accurately modelled. HCPLSR is an extension of Partial Least Squares Regression (PLSR) [19], which has gained popularity in chemometrics [20]. HCPLSR has also previously been used for metamodelling of different biological dynamical systems [8, 21].

Gaussian process regression (GPR) models [22] are probabilistic models which are seeing increased interest with advances in sparse approximations allowing them to be used on much larger data sets [23, 24], and extensions to deep models [25–27]. Since a GPR metamodel provides an explicit distribution that approximates the distribution over simulator outputs, we use the trained GPR metamodels to estimate the posterior distributions over parameters given observed simulation outputs, using MCMC methods. We estimate the posterior distributions separately for the metamodels trained on the population spiking activities and the LFP, for a variety of different simulation outputs. We also investigate some of the properties of the distributions over the domain on which the metamodels were trained, as well as the differences between the posterior distributions generated from the population spiking activities and the LFPs.

2 Methods

2.1 Simulations

The network considered in this study consists of two populations, one with excitatory neurons and one with inhibitory neurons, of size N_E and N_I respectively. The neurons are modelled as leaky integrate-and-fire neurons, connected with alpha-shaped current-based synapses described by Eq 2 in Table 1. The sub-threshold dynamics are described by a first-order differential equation given by Eq 1 in Table 1. When the membrane potential reaches a firing threshold V_{thr}, it fires and and the synapses on all its postsynaptic neurons are activated after a time t_d. After firing, the membrane potential of a neuron is clamped to a voltage V_{reset} for a refractory period of t_{ref} ms.

Each neuron receives synapses from a fraction ϵ of all other local neurons in the network. Post-synaptic currents from excitatory synapses have a peak amplitude J, and post-synaptic currents from inhibitory synapses have a peak amplitude of $-gJ$. Both types of synapses share a synaptic time constant of τ_{syn}. Additionally, each neuron receives independent excitatory input from an external population modelled as a Poisson process with fixed rate ν_{ext}. This rate is determined by the parameter $\eta = \frac{\nu_{\text{ext}}}{\nu_{\text{thr}}}$, where $\nu_{\text{thr}} = V_{\text{thr}} C_m / (J \tau_m e \tau_s)$, the minimum constant input rate that will drive the neuron to its threshold. Since the synaptic strength and synaptic time constant both contribute multiplicatively to the total charge transfer during a synaptic event, we opted to parameterize the synaptic strength in terms of the total charge transfer during a synaptic event $Q_s = \tau_s J$. A complete description of the point-network model is given in Table 1.

Ten model parameters were uniformly randomly sampled, using Latin Hypercube sampling [28], which is a stratified sampling strategy suitable for sampling in high-dimensional systems. A complete list of model parameters and their values and ranges are given in Table 2. 4000 simulations were run using the NEST simulator [29,30], comprising both the training data set and the test data set. The regions of the parameter space giving strongly synchronous firing, discussed in [5,31], were avoided by avoiding unbalanced regimes caused by low inhibition, and high external input ($\eta > 3.5$). Additionally, a small number of simulations, which were found to be strongly synchronous were removed, making up roughly 3% of the total simulations. These were found by the following ad hoc criteria: more than 150 time bins of 1 ms in which more than 10% of the neurons fire, as well as more than 500 time bins in which fewer than 0.25% of the neurons fire. See Section 3.1 and Section 4 for further discussion. For 100 of the simulation outputs in the test data set, an additional 50 simulations were run with parameters sampled from the estimated posterior distribution (see Sections 2.4 and 3.5). For each simulation, the LFP was computed as described in the next section.

The population spiking activities were saved at a resolution of 1 ms as the sum of all spike trains in the populations. The power spectrum, defined as $[\psi](f)^*[\psi](f)$ for some signal $\psi(t)$, where $[\psi](f)$ denotes the Fourier transform of ψ, was estimated using Welch's method [32] for both the population spiking activities and the LFP. The implementation in the Python SciPy [33] package (`scipy.signal.welch`) was used,

with a Hann window of length 256 and an overlap of 128.

Table 1. Description of point-neuron network following the guidelines of [34].

A	**Model summary**
Populations	One excitatory, one inhibitory
Network model	Fixed in-degree, random convergent connections
Neuron model	Local populations: leaky integrate-and-fire, external: Poisson generator
Synapse model	Current-based shaped-shaped, fixed strength for each population
B	**Populations**
Names	Excitatory: E Inhibitory: I
C	**Network model**
Connectivity	Fixed number of incoming connections $C_{\mathrm{E}} = \epsilon N_{\mathrm{E}}$ from excitatory population and $C_{\mathrm{I}} = \epsilon N_{\mathrm{I}}$ from inhibitory population
Input	Poissonian synaptic input with fixed rate ν_{ext} for each neuron
D	**Neuron model**
Type	Leaky integrate-and-fire neuron
Description	Dynamics of membrane potential $V_i(t)$ (neuron $i \in [1, N]$): - Spike emission at times t_l^i with $V_i(t_l^i) \geq V_{\mathrm{thr}}$ - Subthreshold dynamics: $$\tau_{\mathrm{m}} \frac{\mathrm{d}V_i(t)}{\mathrm{d}t} = -V_i(t) + R_{\mathrm{m}} I_i(t) \quad \text{if } \forall l : t \notin (t_l^i, t_l^i + t_{\mathrm{ref}}] \tag{1}$$ where τ_{m} is the membrane time constant, V the membrane potential, R_{m} the membrane resistance, and I the synaptic inputs. - Reset + refractoriness: $V_i(t) = V_{\mathrm{reset}} \quad \text{if } \forall l : t \in (t_l^i, t_l^i + t_{\mathrm{ref}}]$ Exact integration with temporal resolution dt Uniform distribution of membrane potentials $V_i \in [V_{\mathrm{reset}}, V_{\mathrm{thr}})$ at $t = 0$
E	**Synapse model**
Type	Alpha-shaped postsynaptic current
Description	$$R_{\mathrm{m}} I_i(t) = \tau_{\mathrm{m}} \sum_j \frac{J_{ij}}{\tau_s} \sum_l H(t - t_l^j - t_{\mathrm{d}}) t e^{1-(t-t_l^j-t_{\mathrm{d}})/\tau_s}, \tag{2}$$ where the first sum is over all the presynaptic neurons j, including the external ones, and the second sum is over the spike times of those neurons. t_l^j is the lth spike of presynaptic neuron j, and t_{d} is the synaptic delay. H denotes the Heaviside step function. $$J_{ij} = \begin{cases} J, & j \in \{E, E_{\mathrm{ext}}\} \\ -gJ, & j \in \{I\} \end{cases}$$ Multapses and autapses are allowed.

2.2 Forward-model predictions of LFPs

In order to compute local field potentials (LFPs) from the point-neuron network, we utilized the hybrid LFP scheme [35] (github.com/INM-6/hybridLFPy), which allows for the decoupling of the simulation of spiking dynamics (here computed using point neurons) and predictions of extracellularly recorded LFPs. The latter part relies on

Table 2. Point-neuron network parameters.

Point-neuron parameters		
Symbol	Description	Value
η	relative amount of external input	$[1.0, 3.5]$
g	relative strength of inhibitory synapses	$[4.5, 8.0]$
Q_s	total synaptic charge transfer	$[25 - 100]$ fC
τ_{m}	membrane time constant	$[15 - 30]$ ms
C_{m}	membrane capacitance	$[100 - 250]$ pF
t_{d}	synaptic delay period	$[0.1 - 3.0]$ ms
t_{ref}	absolute refractory period	$[0.1 - 4.0]$ ms
τ_{syn}	synaptic time constant	$[1.0 - 8.0]$ ms
V_{thr}	firing threshold	$[15 - 25]$ mV
V_{reset}	reset membrane potential	$[0 - 10]$ mV
E_L	passive leak reversal potential	0 mV
N_{E}	number of excitatory neurons	8000
N_{I}	number of inhibitory neurons	2000
ϵ	connection probability	0.1
C_{E}	number of incoming excitatory synapses	800
C_{I}	number of incoming inhibitory synapses	200
Simulation parameters		
Training and test data		
T_{sim}	simulation duration	10500 ms
$T_{\mathrm{transient}}$	start-up transient duration	500 ms
dt	time resolution	0.1 ms

reconstructed cell morphologies and multi-compartment modelling in combination with an electrostatic forward model. A complete description of the scheme (including the biophysics-based forward model) can be found in [35], and a concise description of the multi-compartment neuron model is given in Table 3. The parameters used are given in Table 4.

Current-based synapses and morphologies with passive membranes in the multi-compartment neuron models give a linear relationship between any presynaptic spike event and contributions to the LFP resulting from evoked currents in all postsynaptic multi-compartment neurons. Thus the LFP contribution $\phi_Y^j(\mathbf{r}, t)$ at position $\mathbf{r}$ from a single presynaptic point-neuron j in population Y can, in general, be calculated by the convolution of its spike train $\nu_Y^j(t) \equiv \sum_k \delta(t - t_j^k)$ with a unique kernel $H_Y^j(\mathbf{r}, \tau)$ as $\phi_Y^j(\mathbf{r}, t) = (\nu_Y^j * H_Y^j)(\mathbf{r}, t)$. This kernel encompasses effects of the postsynaptic neuron morphologies and biophysics, the electrostatic forward model, the synaptic connectivity pattern, conduction delay and post-synaptic currents. This kernel can be further decomposed as a convolution between the LFP response of a unit current for a single time step, and the post-synaptic current. The resulting LFP due to spikes in

Table 3. Description of multi-compartment neuron populations.

A	**Model summary**
Populations	Local excitatory and inhibitory populations
Neuron model	Multi-compartment neurons with passive cable formalism
Synapse model	Current-based α-function shaped, fixed strength for each population
Topology	Cylinder of 1 mm^2 cross-section with somas of both populations positioned in single layer of thickness 0.1 mm.

B	**Neuron models**
Type	Reconstructed multi-compartment morphologies with passive electrical properties
Description	For each neuron, the membrane potential V_n of compartment n connected to m other compartments k, with a surface area a_n, length l_n and diameter d_n is given by: $$\sum_{k=1}^{m} g_{\mathrm{a}kn}(V_k - V_n) = C_{\mathrm{m}n}\frac{\mathrm{d}V_n}{\mathrm{d}t} + I_{\mathrm{m}n} \quad (3)$$ $$C_{\mathrm{m}n} = c_{\mathrm{m}} a_n \quad (4)$$ $$g_{\mathrm{a}kn} = \pi(d_n^2 + d_k^2)/(4r_a(l_n + l_k)) \quad (5)$$ $$I_{\mathrm{m}n} = g_{\mathrm{L}n}(V_n - E_{\mathrm{L}}) + \sum_j I_{jn}\ , \quad (6)$$ where for compartment n, $C_{\mathrm{m}n}$ is the membrane capacitance, $g_{\mathrm{a}kn}$ the axial conductance from compartment k, $I_{\mathrm{m}n}$ the membrane current, $g_{\mathrm{L}n}$ the membrane leak conductance, E_{L} the extracellular reversal potential, and I_{jn} the synaptic current from presynaptic neuron j.

C	**Synapse model**
Synapse type	α-function shaped postsynaptic current
Description	$$I(t) = H(t - t_{\mathrm{a}})JCte^{1-t/\tau_{\mathrm{s}}} \quad (7)$$ $$H(t) = 0 \text{ for } t \leq 0, \text{ otherwise } 1\ . \quad (8)$$ Here t_{a} is the activation time of the synapse, J the synaptic strength, and τ_{s} is the synaptic time constant. C is a constant chosen so that $JC\int_0^\infty te^{1-t/\tau_{\mathrm{s}}}\mathrm{d}t = C_{\mathrm{m}}J$, assuring that the same total charge is transferred as in the δ-function synapse in the point-neuron network.

D	**Topology**
Type	Cylinder with radius $1/\sqrt{\pi}$ mm and height 0.5 mm containing two vertical sections
Description	- Cylinder extends from $z = -500$ μm to $z = 0$ - All somas are randomly placed with a uniform distribution within the boundaries $r \leq 564$ μm and -450 $\mu\mathrm{m} \leq z \leq -350$ μm - Two regions separated by the plane $z = -300$ μm - Synapses on inhibitory neurons are placed in lower region - Inhibitory synapses on excitatory neurons are placed in lower region - Excitatory synapses on excitatory neurons are split equally between regions

Table 4. Multi-compartment neuron parameters.

Multi-compartment neuron parameters		
Symbol	Description	Value
τ_{m}	membrane time constant	20 ms
C_{m}	membrane capacitance	1.0 μF/cm^2
R_{m}	membrane resistivity	$\tau_{\mathrm{m}}/C_{\mathrm{m}}$
R_{a}	axial resistivity	150 Ωcm
τ_{s}	synaptic time constant	5 ms
E_{L}	passive leak reversal potential	0 mV
V_{init}	membrane potentials at $t = 0$ ms	0 mV
σ_{e}	extracellular conductivity	0.3 Sm^{-1}

a presynaptic population Y is then given by

$$\phi_Y(\mathbf{r}, t) = \sum_{j \in Y} (\nu_Y^j * h_Y^j * I_Y^j)(\mathbf{r}, t) \ , \tag{9}$$

where $I_Y^j(t)$ is the post-synaptic current given by Eq 7 in Table 3, and $(h_Y^j * I_Y^j)(\mathbf{r}, t) = H_Y^j(\mathbf{r}, t)$.

As shown in [31, 35], a good approximation can be made by convolving the population firing rates $\nu_Y(t) \equiv \sum_{j \in Y} \nu_Y^j(t)$ with averaged kernels $\overline{H}_Y(\mathbf{r}, \tau) \equiv 1/N_Y \sum_{j \in Y} H_Y^j(\mathbf{r}, \tau)$, that is,

$$\phi_Y(\mathbf{r}, t) = (\nu_Y * \overline{H}_Y)(\mathbf{r}, t) \ . \tag{10}$$

As in [31, 35] these averaged kernels $\overline{H}_Y(\mathbf{r}, \tau)$ were computed using the full hybrid-scheme. This was done by computing the LFP resulting from a fully synchronous activation of all the outgoing synapses from all neurons in the presynaptic population. Thus for the computation of the LFP kernel, we have $\nu_Y^j(t) \equiv \delta(t - t_Y)$ where t_Y is the timing of the synchronous event in population Y.

After the single time-step kernel has been computed, the full kernel can easily be computed for any post-synaptic current.

2.3 Gaussian process regression

Here we give a brief overview of Gaussian process regression models, their training procedures and the extension to deep models.

Gaussian process regression models are probabilistic non-parametric models that provide distributions over arbitrary functions, and are trained using supervised learning.

A Gaussian process (GP) is a collection of random variables such that any finite number of them follow a joint Gaussian distribution, and a GP is completely specified by a *mean function* $m(\mathbf{x})$ and a *covariance function* $k(\mathbf{x}, \mathbf{x})$ [22]. Consider a data set consisting of n inputs of dimension d contained in the matrix X, and corresponding outputs $\mathbf{y}$, which are assumed to be one-dimensional for notational simplicity in this section. A Gaussian process regression model assumes a distribution over an underlying function $f(x)$ which is modelled as a Gaussian process. Evaluating this distribution at the set of input points X gives a joint Gaussian distribution $f(X) \sim \mathcal{N}(\mathbf{0}, k(X, X))$, where the covariance function $k(X, X)$ must return a positive semi-definite covariance matrix between all the data points contained in X. The mean function is usually assumed to be 0 without loss of expressiveness. The covariance function encodes the properties of the function, and there are many different covariance functions available with different properties, e.g. varying degrees of smoothness, periodicity, and stationarity. Different covariance functions can be combined by multiplication and addition, and allows the modeller to integrate domain knowledge into the model. An example of a covariance function is the squared exponential function $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\big(-\frac{(\mathbf{x}-\mathbf{x}')^2}{2l^2}\big)$ where the covariance is determined by the distance between the two input locations, the variance σ^2 and the length scale l, where the latter two are model parameters.

The model of the observed outputs $\mathbf{y}$ is given by the underlying function $\mathbf{f}(X)$ corrupted by some noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 I)$ such that $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$, and follows a distribution $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, k(X, X) + \sigma_n^2 I)$. Given a new set of inputs X_* on which we would like to evaluate the model, the GP specifies a joint distribution over $\mathbf{y}$ and $\mathbf{y}_*$, and the predictions are made as the distribution over $\mathbf{y}_*$ conditioned on the training data $\mathbf{y}$ [22]

$$p(\mathbf{y}_*|\mathbf{y}; X_*, X) = \int p(\mathbf{y}_*|\mathbf{f}_*; X_*) p(\mathbf{f}_*|\mathbf{f}; X, X_*) p(\mathbf{f}|\mathbf{y}; X) d\mathbf{f} d\mathbf{f}_* \tag{11}$$

$$= \mathcal{N}(\mathbf{y}_*|K_{n*}K_{nn}^{-1}\mathbf{y}, \quad K_{**} - K_{f*}^T (K_{nn} + \sigma_n^2 I)^{-1} K_{f*} + \sigma_n^2 I), \tag{12}$$

where we use the shorthand notation $K_{nn} = k(X, X)$, $K_{n*} = k(X, X_*)$ and $K_{**} = k(X_*, X_*)$. The semicolon indicates the input points corresponding to the distributions, on which the covariance function is evaluated. The marginal likelihood of

the training data is obtained as

$$p(\mathbf{y}; X) = \int p(\mathbf{y}|\mathbf{f}; X)p(\mathbf{f}; X)d\mathbf{f} = \mathcal{N}(\mathbf{0}, K(X, X) + \sigma_n I), \tag{13}$$

which can be found analytically in the case of a Gaussian likelihood $p(\mathbf{y}|\mathbf{f}, X)$, and is used to optimize the model parameters.

Although the above framework is elegant, it has a few drawbacks. First, if the likelihood $p(\mathbf{y}|\mathbf{f})$ is not Gaussian, the integrals in eqs. 11 and 13 do not have closed-form solutions. Second, an $n \times n$ matrix must be inverted to compute both the marginal likelihood in Eq 13 and the predictive posterior in Eq 11. This scales as $\mathcal{O}(n^3)$, and is prohibitively expensive for large data sets. Approximate methods must be used to overcome both of these problems.

Methods to reduce the complexity have been extensively studied [22, 36], and typically make use of *inducing points*, a set of m additional input points Z with corresponding distributions $\mathbf{u}$ under the same GP prior as $\mathbf{f}$, where $m \ll n$, and Z is in the same space as X. The GP specifies a joint distribution over both $\mathbf{f}$ and $\mathbf{u}$, and a distribution over $\mathbf{f}$ can be inferred as

$$p(\mathbf{f}; X) = \int p(\mathbf{f}|\mathbf{u}; X, Z)p(\mathbf{u}; Z)d\mathbf{u}.$$

By introducing an additional free-form Gaussian distribution over $\mathbf{u}$, parameterized by the mean $\mathbf{m}_u$ and covariance matrix $S_{\mathbf{uu}}$, the model parameters, along with the variational parameters $\mathbf{m}_u$ and $S_{\mathbf{uu}}$, are optimized using variational inference. Evaluating the model is done using the variational distribution over $\mathbf{u}$ instead of the training data as in the standard GP model. Since m, the number of inducing points, is assumed to be much smaller than n, the number of examples in the training data set, this method is called a *sparse* approximation. The optimization of sparse GP models is mathematically more complex than the regular case, and we do not go into the details here, but refer the reader to [38] for an excellent tutorial.

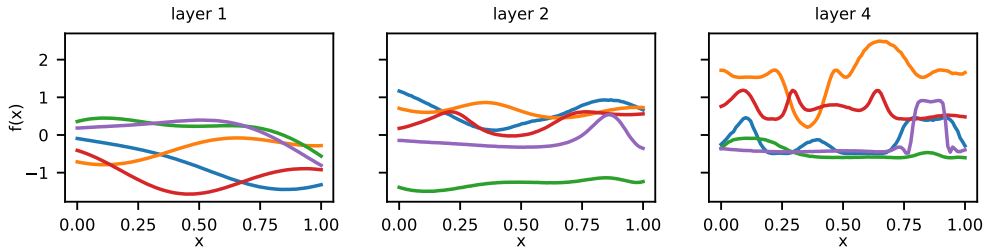

Fig 1. Five examples from functions drawn from a 4-layer GP, with the output at layers 1, 2 and 4 shown. The five functions have the same colour in all three plots. Apart from layer 1, each layer takes output of the previous layer as input.

Deep Gaussian processes

The above section describes a model in which a single GP, with it's associated covariance and mean function, describes the mapping from inputs X to outputs $\mathbf{y}$. This model is still limited in its expressiveness by the covariance function. Although covariance functions can be tailored to specific tasks, it is a time-consuming process, and complex functions with many model parameters run the risk of overfitting. Additionally, the function may have different properties at different locations in the input space, which have to be explicitly represented by change points, possibly at multiple unknown locations.

Deep GP models provide an alternative way of expanding the model expressiveness by adding successive layers of functions with the previous layer as inputs. With L layers, a deep GP recursively defines a distribution over all the layers as

$$p(\mathbf{y}, \{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) = \prod_{i=1}^N p(y_i|f_i^L) \prod_{l=1}^L p(\mathbf{f}^l|\mathbf{u}^l; \mathbf{f}^{l-1}, Z^{l-1}) p(\mathbf{u}^l; Z^{l-1}) \ ,$$

where $\mathbf{f}^0$ is taken to be X [26]. Each layer contains separate inducing points as in sparse GPs. Inference in this model is intractable, so approximations must be made. Multiple independent GPs can be deployed in each layer, in which case the GPs in the next layer would take all of them as input. Figure 1 shows a few examples of functions drawn from a deep GP with 4 layers. In the first layer, the function has uniform properties. As the number of layers increase, the complexity of the function increases, and it is no longer uniform in its properties.

Originally introduced by [25], multiple inference schemes have been proposed since [26, 27], where we use the implementation proposed by [26] since it does not

assume independence between layers and due to its simplicity of implementation in packages such as GPyTorch [37]. We do not go into details about the inference procedure here, but refer to [26, 38] for the full details.

Linear model of coregionalization

When dealing with functions with multiple outputs, one can either assume independence between output dimensions, or include correlations between them. With a large number of output dimensions, having a separate independent GP for each dimension is costly. The linear model of coregionalization (LMC) offers a simple way of constructing correlated outputs based on a smaller number of independent GPs by constructing the full output as a linear combination of independent GPs [39].

The model can be written as

$$\mathbf{g}(\mathbf{x}) = \{g_p(\mathbf{x})\}_{p=1}^{P}, \qquad \mathbf{f}(\mathbf{x}) = W\mathbf{g}(\mathbf{x}), \tag{14}$$

where $g_p(\mathbf{x})$ forms a set of independent GPs, and W is an $n \times p$ matrix used to linearly construct the output $\mathbf{f}(\mathbf{x})$. This can be viewed as an input-dependent linear latent space model, where we assume a linear mapping between a latent space and the output, but the coefficients of this mapping can change in a non-linear fashion. If we choose P to be a small number, a low-rank representation of the output is achieved.

We used a deep GP based on the framework by [26] with 2 layers, where the first layer has 10 independent GPs, the second layer 14 independent latent GPs, and the final output is given by an LMC, i.e. a linear combination of the 14 independent GPs in the second layer. We implemented the model in GPyTorch v.1.6.0 with PyTorch v1.9.0 as backend.

2.4 Posterior distribution over parameters

The GPR framework described above provides a distribution $p(\mathbf{x}|\boldsymbol{\theta})$, which is an approximation to the distribution over the outputs given by the simulations, where $\mathbf{x}$ is the simulation output, and $\boldsymbol{\theta}$ is the simulation inputs. Introducing a prior distribution over the inputs $p(\boldsymbol{\theta})$, which we take to be uniform over the ranges of input values used to train the metamodels, the unnormalized posterior distribution over parameters given

a specific observed model output $\mathbf{x}_0$ is given by $p(\boldsymbol{\theta}|\mathbf{x}_0) \propto p(\mathbf{x}_0|\boldsymbol{\theta})p(\boldsymbol{\theta})$. We sampled from this distribution using an adaptive Metropolis algorithm [40]. The adaptive Metropolis algorithm is a random walk Markov Chain Monte Carlo algorithm, where a multivariate Gaussian proposal distribution is used, and where its covariance matrix is iteratively estimated by the sample covariance. We only estimated the covariance during the burn-in period, and kept it constant during sampling. Additionally, we used a separate scaling factor for the covariance matrix which was tuned during the burn-in period to give an acceptance ratio of around 0.3. Five chains were run in parallel, and run for a total of 40000 evaluations of the proposals, where the samples from the first 12000 proposals were discarded. For each example in the test data set, the posterior distribution was sampled separately for both the metamodel of the population spiking activities and the metamodel for the LFP. The Gelman-Rubin statistic [41] was computed independently for each parameter, for each distribution, in order to assess the convergence properties of the chains.

2.5 Hierarchical Cluster-based Partial Least Squares regression

In this section, we give a brief overview of the Hierarchical Cluster-based Partial Least Squares regression (HCPLSR) method for metamodelling. HCPLSR is an extension of regular Partial Least Squares regression (PLSR). Given a $n \times d$ matrix of normalized independent variables X, and a $n \times p$ matrix of normalized response variables Y, PLSR finds a low-dimensional representation of the features, and applies a linear multivariate regression from the latent space to the response variables. The independent and response variables are decomposed as described in [20, 42]

$$X = T_A P_A^T + E_A,$$
$$Y = T_A Q_A^T + F_A,$$

where T_a is an $n \times l$ matrix called the *x-scores*, P_A and Q_A are $d \times l$ matrices called the *x-loadings* and *y-loadings* respectively, and E_A and F_A are residuals. The subscript A denotes the number of latent dimensions, i.e. the rank of the approximation, such that $l = A$. In an iterative procedure, a column of the scores $T_a,\ a = 1, 2, .., A$, are found as the projection of X_a onto the left-singular vector of $X_a^T Y_a$ corresponding to the largest

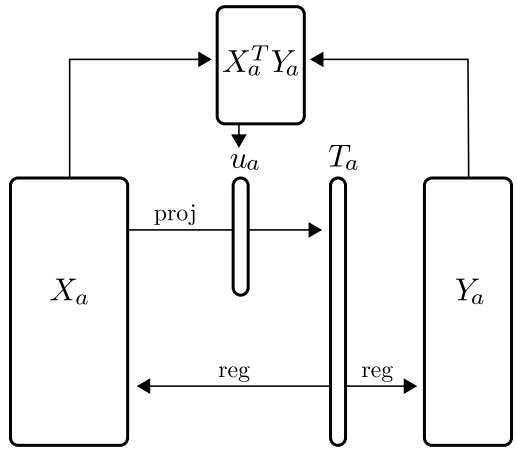

Fig 2. Schematic of PLSR training procedure. In each iteration a of the procedure, the left-singular vector of $X_a^T Y_a$, is computed, denoted by u_a. The x-scores T_a are computed as the projection of X_a onto u_a, and the matrices X_a and Y_a are deflated by the linear regression of T_a onto X_a and Y_a respectively.

singular value. Linear regression from T_a to X_a and Y_a is performed with regression coefficients $\mathbf{p}_a$ and $\mathbf{q}_a$ respectively, and X_a and Y_a are deflated according to $X_{a+1} = X_a - T_a \mathbf{p}_a^T$ and $Y_{a+1} = Y_a - T_a \mathbf{q}_a^T$. The procedure is iterated A times, determined by how many components one wishes to use, with an upper limit of $\text{rank}(X)$. X_1 and Y_1 are taken to be the normalized data matrices X and Y respectively. Figure 2 shows a diagram of a single iteration in the procedure. Predictions are made by $y_{\text{pred}} = T_A Q_A^T$.

In HCPLSR, multiple PLSR models are deployed in different regions of the parameter space, increasing the flexibility of the model. A global PLSR model is first applied, on which fuzzy C-means clustering (FCM) [43, 44] is performed on either the x-scores or y-scores. A separate local PLSR-model is subsequently trained on each cluster, and the final prediction is made by a weighted sum of the local model predictions, with the weights given by the probability of belonging to the separate clusters as determined by the fuzzy clustering. Alternatively, the prediction can be done by choosing the local model made within the cluster that the observation has the highest probability of belonging to. The number of clusters to use, the number of latent components in the global PLSR model and the number of latent components in the local PLSR models must be set by the user, based on an exploration of a number of different model alternatives. Additionally, the fuzzifier parameter of the FCM algorithm, which can be interpreted as the degree to which points are shared among the different clusters, must be specified, which was here chosen to be 2 as in [18]. For a full

description of the HCPLSR model, we refer to [18]. In this study, we use a HCPLSR model based on the PLSR implemention in the Python scikit-learn package [45] (`sklearn.cross_decomposition.PLSRegression`), with a second degree polynomial basis expansion of the inputs, i.e. expanding the input by the squares and second order cross terms of the parameters. A grid search was done over the number of global and local components, as well as the number of clusters, and the model with the lowest mean error over the test data set was chosen.

2.6 Technical details

Reproducibility

The simulations were run using Python v3.8.5. All point-network simulations were run with the NEST simulator v2.20.0 [29, 30]. The forward-modeling of the LFP was done using hybridLFPy v0.1.4 [35] , with NEURON v7.7 [46]. The GPR metamodel was trained using GPyTorch v1.6 with PyTorch v1.9. The simulations and training of the GPR metamodels were run on Intel Xeon Gold 6146 CPUs. Evaluations of the metamodels were run on an Intel Core i5-8265U CPU. All simulation and analysis code for this study is available on Github:

3 Results

The aim of this study was to apply metamodelling techniques to simulations of neuronal networks. Specifically, we applied GPR and HCPLSR metamodels on the power spectra of both the population spiking activities and local field potential of a two-population network of point-neurons, and compared them to ground truth simulations. Since the GPR metamodel gives an explicit estimation of the model likelihood, i.e. the distribution over the power spectra given model parameters, it can be used to estimate the posterior distribution over parameters given an observed power spectrum. Figure 3 shows an overview of the modelling scheme.

We start by showing some example model outputs, and discuss the accuracy of the forward-predictions of the GPR metamodel. We then compare the performance of the GPR metamodel to the HCPLSR metamodel and evaluate the number of training

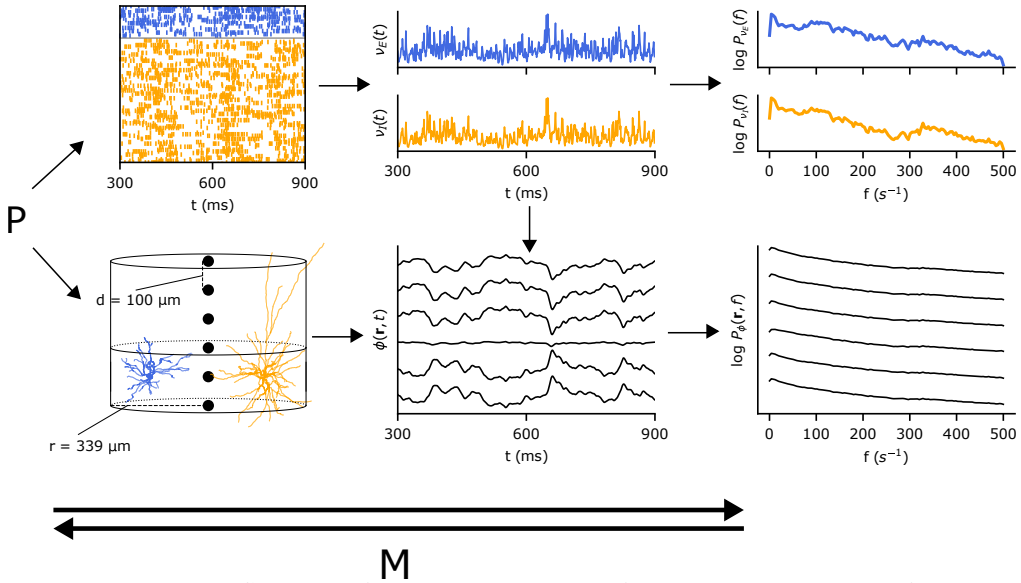

Fig 3. Model overview: Given a set of parameters P, a network of point-neurons are simulated, from which the population spiking activities $\nu(t)$ are computed (top). With morphological neurons spatially extended in a column, the synaptic spiking activities from the point-neuron network can be 'replayed' on the morphological neurons, and the resulting local field potential $\phi(\mathbf{r}, t)$ at 6 locations (black dots in column) is computed (bottom). Note that only one excitatory and one inhibitory neuron is shown, in simulation there is one for each point-neuron. The power density spectra $P_\nu(t)$ and $P_\phi(\mathbf{r}, t)$, of the population spiking activities and local field potential respectively, are also computed. The metamodel M directly models the power spectra given the parameter set, or the parameters given observed power spectra.

samples needed to obtain a metamodel giving high prediction accuracy, before we discuss the inverse modelling and the posterior distributions over parameters.

3.1 Simulations

A total of 4000 simulations were run, with parameters sampled as described in Section 2.1. A network consisting of one excitatory and one inhibitory population can produce a variety of activity states, characterized by whether the neurons fire in *synchrony*, and how *regularly* they fire, i.e. whether the inter-spike intervals for individual neurons have a wide or narrow distribution. The behaviour of the present two-population network was analyzed in the framework of mean-field theory in [5] for integrate-and-fire neurons where the synaptic currents were modelled as delta functions. Brunel [5] found that the network spiking activity could be described by distinct regimes, classified as synchronous regular (SR), asynchronous irregular (AI) and synchronous irregular (SI, fast and slow), depending on which specific parameter values were used, and arise by different mechanisms. As described in Section 2.1, we avoided activity states characterized by strongly synchronous behaviour, which produces sharp peaks in the

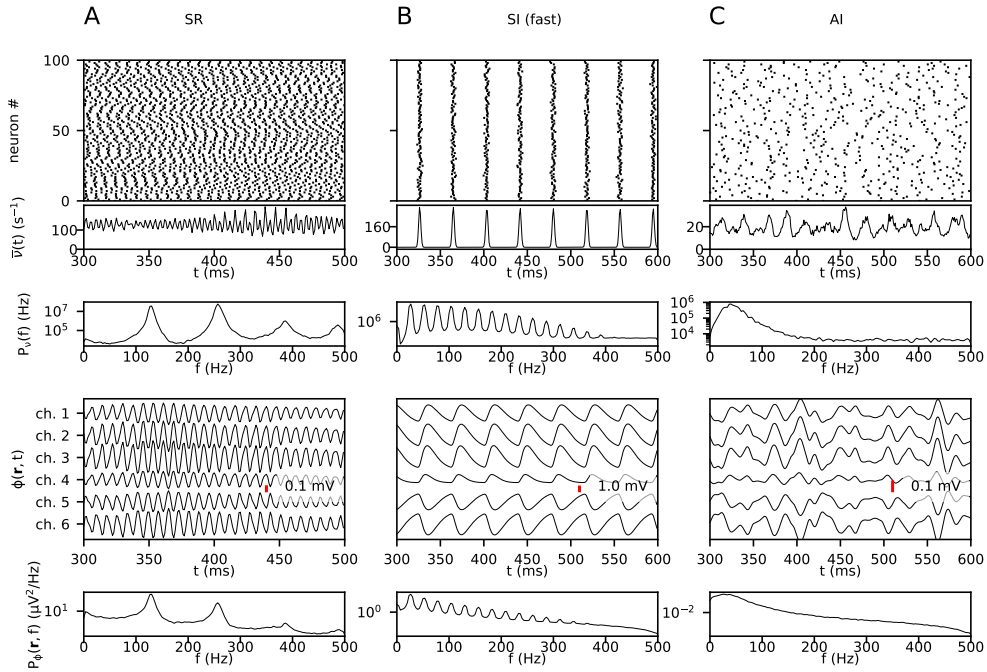

Fig 4. Example activity states. Each column shows, from top to bottom, a raster plot , population firing rates, the power spectrum of the population firing rates, the LFP, and the power spectrum of the LFP). A: Synchronous regular activity. B: Synchronous irregular activity. C: Asynchronous irregular activity.

power spectrum. Figure 4 shows three examples of different states of network activity. Column A shows the synchronous regular regime, characterized by runaway excitation, where the synaptic strengths of inhibitory neurons are not strong enough to overcome the excitation, and the neurons fire regularly and synchronously with very high rates. Column B shows an example of a synchronous irregular state, produced in some cases when the network receives high amounts of external input and the inhibitory synaptic strengths are high. Column C shows asynchronous irregular activity. This is the regime in which most of the simulations used for training and validation were, although there is of course much variation within this regime. Columns A and B show the type of activity we avoided, with sharply peaked spectra.

We will not attempt to give a full description of all possible network states or the effects of all different parameters in this paper, but show these examples to make clear the states we have actively avoided, as described in Section 2.1.

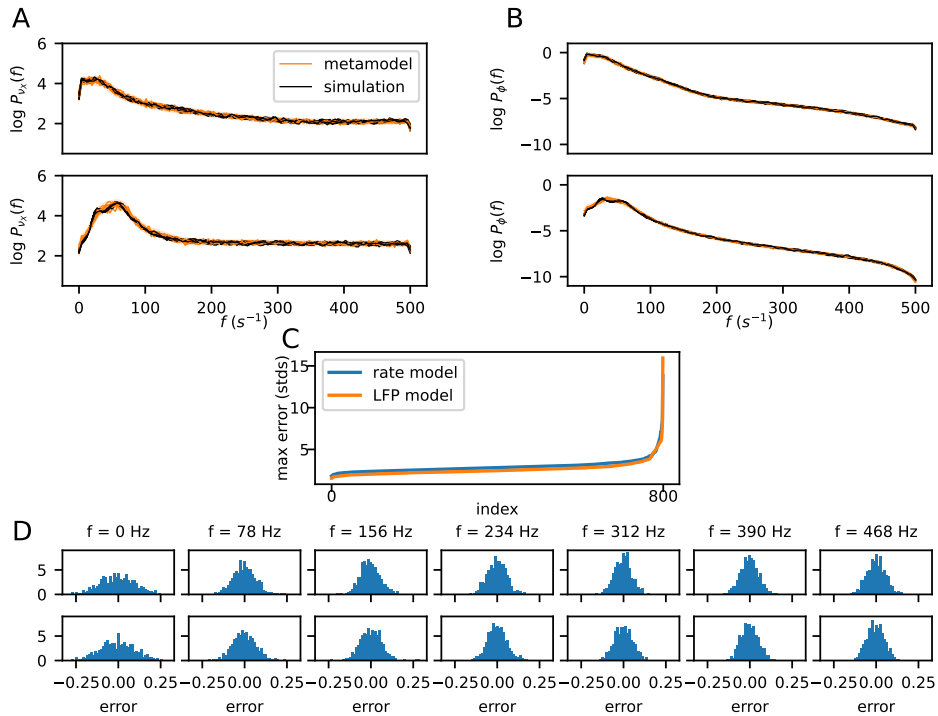

Fig 5. A: Each plot shows 10 example simulation outputs (black lines), and 10 random examples drawn from the probability distribution given by the GPR metamodel (orange lines) of the power spectrum of the excitatory population spiking activities for two different parameter sets. The shaded orange area shows two standard deviations above and below the mean of the distribution over the output. The top and bottom example correspond to the 10th and 90th percentile of maximum errors across all frequencies respectively. B: Same as A, but for the topmost channel of the LFP. C: Sorted maximum errors over all frequencies and channels for each example in the test data set, . Errors are shown in standard deviations of the distribution given by the metamodel. D: Distribution over errors, when compared to ground truth, in absolute scale for a few different frequencies for the population spiking activities (upper) and LFP (lower).

3.2 Forward predictions

The metamodels were tested on a data set containing 800 examples, jointly sampled with the training data using Latin Hypercube sampling. Models were trained on data sets ranging from 100 to 3000 examples in order to examine the density of samples needed to adequately represent the simulation output space. Figure 5A and B shows examples of simulated outputs and corresponding outputs from the GPR metamodel for two different sets of parameter values, for the population spiking activities and LFPs respectively. Only the output for the excitatory population spiking activity and the topmost LFP channel are shown. 10 simulations run with different seeds are shown in black lines, and 10 samples drawn from the output distribution given by the GPR metamodel in orange lines. The shaded orange area shows two standard deviations above and below the mean. The example shown on the top plot corresponds to the 10th percentile in errors, taken as the maximum across all frequencies, and the bottom plot corresponds to the 90th percentile. In the top example, the metamodel accurately captures both the mean and the variance of the simulated examples for all frequencies for both the population spiking activity and the LFP. In the bottom example, there is a larger discrepancy between the metamodel and the simulator for the lower frequencies for the population spiking activities, but not the LFP. In both of the examples shown here, the smoothness of the power spectra is underestimated for lower frequencies, as can be observed by comparing the individual orange and black lines. Figure 5C shows the sorted absolute maximum errors in standard deviations of the metamodel, where the maximum is taken over all frequencies. We chose the maximum error over frequencies since the large number of output channels make the mean small even in the presence of a few large deviations. Moreover, as the simulation output itself is random, we are mainly interested in finding any larger deviations. While for the bulk of the parameter space, the errors are low and at a similar scale, there are a few examples where the model fails significantly. These are simulations in the strongly synchronous regime that are still present in the data set, after removing most of them as described in Section 2.1. The predictions are unbiased for all frequencies, a few of which are shown in Figure 5D for the population spiking activities (top) and the LFP (bottom).

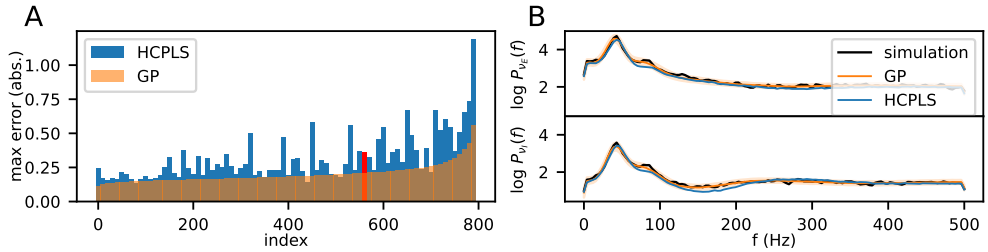

Fig 6. A: Maximum error, in absolute scale, over frequencies on the test data set of population spiking activities, sorted by increasing errors for the GPR metamodel. In order to make the lines visible, only every 10th index is shown. The red bar shows the error corresponding to the example shown in B. B: Example predictions by the GPR metamodel (orange line) and the HCPLSR metamodel (blue line) compared to simulation (black line). The errors of this example corresponds to the red bar in A.

3.3 Comparison to HCPLSR model

We next compare the Gaussian process metamodel to a HCPLSR metamodel. Fig 6A shows the maximum absolute errors, in absolute scale, by the metamodels, where the maximum is taken over all frequencies. The indices are sorted by the errors of the GPR metamodel, and only every 10th index is shown. In nearly all the test examples, the GPR metamodel performs better than the HCPLSR metamodel, but the difference is typically small. Fig 6B shows a simulation output (black lines) along with the predictions by the GPR metamodel (orange lines) and the HCPLSR metamodel (blue lines), for a typical example in terms of errors. The error corresponding to the example shown, is indicated by the red bar in A. In terms of training and evaluating the metamodels, the HCPLSR model is significantly faster, and can be trained in a a few seconds. In contrast, the GPR model takes on the order of a few hours to train. Evaluating the GPR model at 100 different input locations takes around 1.5 seconds, while the same evaluation of the HCPLSR model takes a few milliseconds. The HCPLSR model also gives a set of latent variables which can be useful for interpretation. In contrast to the GPR model, only point estimates of the output are provided.

3.4 Number of training samples needed to represent the model

Both metamodels were trained on data sets ranging from 100 to 3000 examples of the population spiking activities, and the performances on the same test data set were measured. Figure 7A shows the maximum prediction errors over frequencies, averaged over the test data set for each trained metamodel. As the number of training samples increases, the prediction errors on the test data set decreases, as the metamodel starts

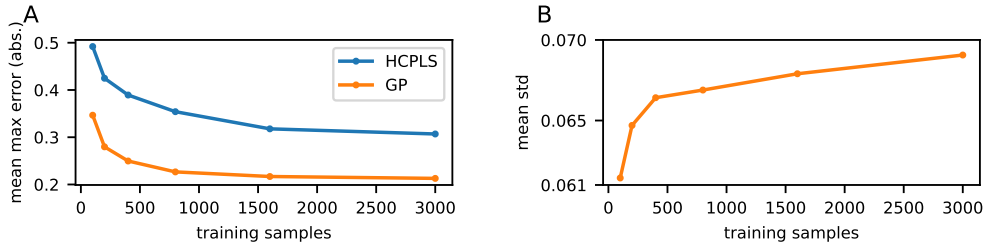

Fig 7. A: Mean (over test data set) of the maximum (over frequencies) errors, in absolute scale, as a function of the number of examples used to train the metamodels on the population spiking activities. B: Mean (over test data set and frequencies) of the standard deviation of the GPR metamodel output.

to generalize better. After a sharp decrease in errors as the number increases to a few hundreds, the benefit of increasing the number of training samples decreases significantly after on the order of 1000 is reached. The same applies for both the GPR and the HCPLSR metamodels. Figure 7B shows the mean, taken over all test examples and frequencies, of the standard deviations of the GPR metamodel output. As multiple examples in the same region may be needed to accurately estimate the variance of the output, the variance also increases with the number of training examples, and similarly to the errors, the effect of increasing the number of training examples diminishes significantly after reaching on the order of 1000 samples.

3.5 Inverse modelling

Using the conditional distribution over simulation outputs given parameter values, estimated by the GPR metamodel, a posterior distribution for each simulation in the test data set was estimated. Separate posterior distributions for both the metamodel of the population spiking activities and the metamodel of the LFP were estimated using an an adaptive Metropolis algorithm as described in Section 2.4. The Gelman-Rubin statistic was computed for all the distributions, and was below 1.1 for over 90% of the posterior distributions for the population spiking activities, and over 95% of the posterior distributions for the LFP, indicating that the chains converged for most of the distributions. In the computation of average values over the test data set, all distributions were used, regardless of the value of the Gelman-Rubin statistic. For 100 of the estimated posterior distributions of both the population spiking activities and the LFP, 50 additional simulations were run with parameters drawn from the posterior distributions.

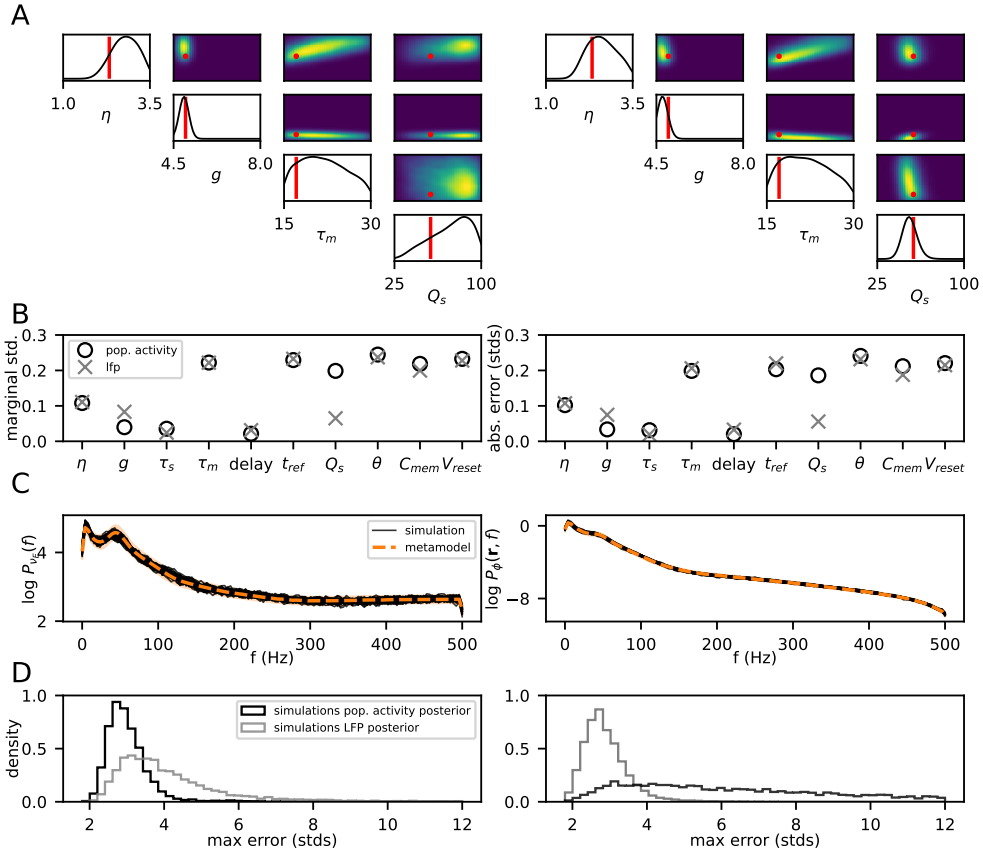

Fig 8. In A, B and D, the left-hand and right-hand side show equivalent plots for the population spiking activities and LFPs respectively. A: 1D and 2D marginal posterior distributions over a subset of the parameters, for an example in the test data set. The red dots and bars show the parameter values corresponding to the simulation output the posterior distribution was computed for. B: Left: mean (across test data set) of standard deviation of the 1D marginal distributions for each parameter. Right: Absolute error of the parameter predictions based on the expectation. C: Metamodel predictions for the power spectra of excitatory population spiking activity (left) and the uppermost channel of the LFP (right) in orange. Black lines show 50 simulation outputs run with parameters drawn from the posterior distribution. D: Distribution of maximum distance between the metamodel mean (orange dashed line in B) and the simulation output from the posterior distributions (black lines in B), in standard deviations of the metamodel output distribution.

Figure 8A shows an example of a posterior distribution generated by the outputs of one simulation in the test data set. The 1D and 2D posterior marginal distributions of a subset of the parameters are shown. On the left-hand side, the posterior distribution generated by the population spiking activities are shown, and the corresponding posterior distribution from the LFP is shown on the right-hand side. The red bars and dots show the values of the parameters used in the simulation from which the posterior is generated. Qualitatively, the distributions from both the population spiking activities and the LFP appear similar for the first three parameters shown.

For this particular example, the 1D marginal distributions of both posteriors are wide for the membrane potential τ_{m}, quite narrow for g, and somewhere in between for η. Wide marginal distributions entail that, lacking information about all other parameters, the value of the parameter cannot be determined with certainty, while narrow marginal distributions entail that the parameter can be accurately determined even without knowledge of other parameters. There is a clear positive correlation between η and τ_{m} in the posterior distribution, i.e. if the value of η is increased the value of τ_{m} must also be increased for the model to produce similar outputs. Equivalently, if you know the specific value of either η or τ_m, the uncertainty in the other parameter would decrease significantly.

Note that because we only see 1D and 2D marginals, there may be higher-order interactions present that cannot be easily visualized, as well as interactions involving the remaining six parameters not shown in this plot. For the synaptic strengths, the distribution is much wider for the model trained on the population spiking activities compared to the model trained on the LFP.

The left-hand side of Fig 8B shows the mean (over all test examples) standard deviations of the 1D marginal distributions. Generally, the parameters related to the synaptic inputs have the narrowest marginal distributions, while the neuron parameters have much wider marginal distributions, indicating that changes to neuron parameters can largely be compensated for by adapting other parameters, while changes to synaptic and input parameters have a larger effect that cannot be compensated for to the same degree. The right-hand side of Fig 8B shows the mean (over all examples in the test data set) absolute distance in standard deviations of the parameters of the simulation used to generate the posterior (i.e. red dots in Fig 8A) and the expectation of the

posterior distribution. For all the parameters, the mean absolute distance is smaller than the standard deviation of the approximate marginal posterior distributions, indicating that the approximate posteriors generally covers the parameters of the simulation used to generate the posterior distribution.

One notable difference between the posterior distributions computed from the metamodel of the population spiking activities and the posterior distributions computed from the metamodel of the LFP is that the overall synaptic strength, Q_s, is significantly more constrained by the LFP than by the population spiking activities, with a standard deviation more than twice as large for the population spiking activities. This can also be observed in the example distributions shown in Fig 8A. For the rest of the parameters, the standard deviations of the 1D marginal distributions are roughly the same when computed for the population spiking activities and the LFP.

Fig 8C shows the metamodel output (orange) corresponding to the same simulation as in A, and 50 individual simulation outputs (black) run with parameter values sampled from the posterior distribution of the example shown in A. Only the spectra from the excitatory population spiking activity and the topmost LFP channel are shown. The mean of the metamodel distribution is shown in dashed orange, and the shaded area shows $\pm$ 2 standard deviations. The simulations run with parameters from the posterior distribution fit nicely with the distribution over the outputs given by the metamodel, suggesting that the posterior distribution is reasonable.

Fig 8D shows the distribution of the maximum (taken over frequencies and simulations for a given posterior) distances (in standard deviations) of the simulation outputs run with parameters drawn from the posterior distributions, and the metamodel mean, i.e. the largest distance between any of the black lines and the orange dashed line for the example shown in Fig 8C, for the 100 posterior distributions for which the extra simulations were run. The black line shows the distribution for the simulations run with parameters drawn from the posterior generated by the population spiking activities, while the gray distribution shows the distances for the simulations run with parameters drawn from the posterior generated by LFPs. The left-hand side shows the distances of the population spiking activities and the corresponding metamodel, while the right-hand side shows the distances for the LFP and the corresponding metamodel. Simulations run with parameters drawn from the posterior distributions have similar simulation outputs

for the same signal from which the posterior distribution was generated, i.e. simulations with parameters drawn from a posterior distribution computed from the population spiking activities have similar population spiking activities, and likewise for LFPs. For the posterior distributions generated by the LFP, the simulations do not give as similar population spiking activities, but they are reasonably close (gray line on the left plot). For the posterior distributions generated by the population spiking activities, however, the simulations give LFPs that are significantly more dissimilar (black line on the right plot), which again indicates that the LFP constrains the parameters to a higher degree.

The Pearson correlation coefficient was computed for all pairwise combinations of parameters for all sampled posterior distributions. Fig 9 shows the mean of the correlation coefficients over all examples in the test data set. The upper triangle shows the coefficients for the samples from the posterior distribution from the population spiking activities, and the lower triangle shows the coefficients for the LFP. The correlations between most of the parameters are fairly weak, but since only the average is shown, there could be stronger correlations in individual posterior distributions that the plot does not capture. For the LFP, there are strong average positive correlations between the synaptic strength Q_{s} and the synaptic time constant τ_s and relative inhibitory strength g. Interestingly, the corresponding coefficients for the population spiking activities are close to 0. This again highlights the difference in the posterior distributions for Q_{s} between the population spiking activities and the LFP. Other strong average correlations are present between the parameters η and τ_{m}, which are present for both the LFP and the population spiking activities. As these parameters all influence the amount of input it takes for the neurons to fire, the correlations and their signs make intuitive sense. For the LFP only, there is additionally a negative correlation between η and C_{mem} not present for the population spiking activities.

4 Discussion

In the present work we have trained metamodels to estimate the power spectra of population spiking activities and local field potentials generated by a two-population neuronal network comprising an excitatory and an inhibitory population during stationary activity. Specifically, we have trained a deep Gaussian process regression

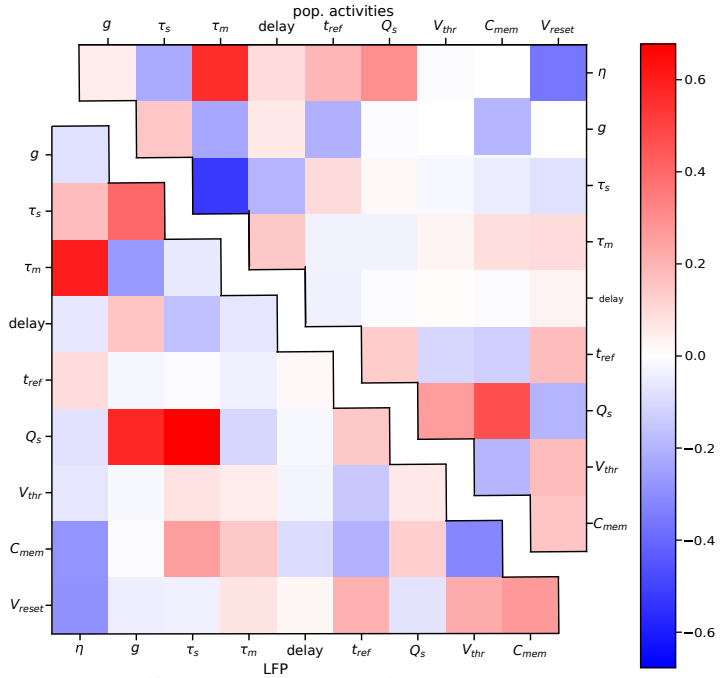

Fig 9. Pearson correlation coefficients between samples from posterior distribution over parameters, averaged over the test data set. Upper triangle shows the correlation coefficients for the posterior distributions from the simulated power spectra from the population spiking activities, while the lower triangle shows the same correlation coefficients from the LFP.

model, and compared it to a much faster HCPLSR model, which has previously been used for metamodelling of biological dynamical systems [18, 47], and used the probabilistic nature of the GPR model to estimate posterior distributions over parameters given observed simulation outputs.

Both the population spiking activities and the LFP were modelled, as this offers an opportunity to compare the information about the parameters that can be gained by observing different signals. The LFP was computed separately for each population using Eq 10. This is an approximation that does not require computing the LFP response from each individual neuron in each simulation, which is computationally costly, but only requires computing a kernel once, which can then be used to approximate the LFP for all simulations. The synaptic parameters can be fully adjusted for in this kernel, but the neuron parameters cannot be changed without computing a new kernel. There is therefore a discrepancy in some of the parameters between the point-neuron network generating the activity and the network of morphological neurons used to generate the kernel. Doing a full simulation of the LFP for each simulation may have resulted in a 'richer' signal with more information about the parameters.

The neuron and synapse parameters of the simulations were varied, with values described in Table 2, while the number of neurons and number of synapses were fixed. The parameter values were chosen such that the strongly synchronous regions of the parameter space were avoided, particularly avoiding relative inhibitory synaptic strengths that lead to an unbalanced regime. Additionally, a small number of simulations in the strongly synchronous irregular regime were removed, comprising roughly 3% of the total number of simulations as described in Sections 2.1 and 3.1. The simulations in the test and training data sets consisted mostly of simulations in the asynchronous irregular (AI) regime. The simulations with strongly synchronous spiking activities were avoided since sharp peaks that shift continuously in frequency cannot be well approximated by a low-dimensional representation, and the low-rank approximation was necessary to keep the size of the GPR metamodel small. The low-rank approximation is a limiting factor for the metamodels, and ideally one would not have to limit the input domain in such a way. Increasing the dimensionality of the latent space is possible and would likely also have resulted in better accuracy, particularly with a larger training data set, but trade-offs have to be made with respect to the computational cost of training and evaluating the metamodels. The simulations in the AI regime could be well approximated by a low-dimensional representation, and since one is often particularly interested in this regime, we consider it reasonable to focus the metamodels on this.

The power spectra were modelled instead of the time series. Although modelling time series directly in many cases would be preferable, that would require modelling the complex temporal dependencies in the neural networks, which are also stochastic due to the external input to the network modelled as a Poisson process. Since most regression techniques assume a fixed target, modelling the power spectra is more straightforward. For the present situation, we found that both the GPR and HCPLSR metamodels were able to accurately predict the power spectrum of the population spiking activities, with 10 different model parameters as inputs, and we found that on the order of 1000 examples is sufficient to accurately represent the model in this particular region of the parameter space. In general, the number of examples needed to train a metamodel will depend on the number of parameters included, the range of their values and how much they affect the model output, and is difficult to predict a priori, and the presently used

model may not be representative for other, more complex models, which may require more training examples.

The forward modelling of the population spiking activities and the LFP is similar in terms of difficulty, where the errors of the GPR metamodels are similar on the scale of the metamodel output distributions. This may not be surprising, as the LFP is computed as a convolution of the population spiking activities. The low-pass filtering effects of synaptic and dendritic filtering makes the power spectrum of the LFP smoother than the population spiking activities, however, which could potentially have made it an easier target [48, 49]. Comparing the GPR metamodel to the HCPLSR metamodel, we found that the GPR metamodel provided better accuracy, but comes at the price of a higher cost of both training and execution. The errors for both metamodels were for many of the examples in the test data set similar, where the GPR metamodel only performed slightly better.

Model inversion

One key motivation for using probabilistic metamodels is that it allows inverting the model and finding posterior distributions over parameters given observed outputs. We opted to sample the posterior using an adaptive Metropolis algortihm as it is simple to implement and worked well for the present problem. The Gelman-Rubin statistic was computed, and for most of the sampled posterior distributions ($> 90\%$), it was below 1.1, which is often considered to indicate convergence [50]. For some of the posterior distributions, the chains did not converge, although this was likely only for a minority. Particularly for the examples on which the forward predictions are poor, we would not expect the estimated posterior distributions to be accurate, as the metamodel generally will not produce outputs which are similar to the observed output used to generate the posteriors.

The degree to which the different model parameters were constrained by observing an output varied highly (see Figure 8B). Notably the distribution over neuron parameters were wide, while the distribution over synapse and input parameters were narrower. Interestingly, the synaptic strength (Q_s) was significantly more constrained by the LFP than by the population spiking activities, illustrating the utility of using

different signals for model validation. The effect is likely due to the fact that LFP is directly dependent on both the post-synaptic currents as well as the the population spiking activities. The effect was not seen for the synaptic time constant, which also directly influences the post-synaptic current, something that indicates that the extra information may be in the scale of the LFP, as that is directly determined by the scale of the post-synaptic currents. For the remaining parameters, the parameters were similarly constrained by both signals, indicating that for the present situation, both signals may contain similar information about the parameters.

A previous study on model inversion using the LFP of a similar network using convolutional neural network [31] provided more accurate predictions of parameter values. In that study, neuron models with delta synapses were used, and it was limited to only three parameters, which likely makes it an easier task. Particularly, wide marginal posterior distributions might only be possible if there are sufficiently many parameters that are free to vary such that they can compensate for the effects of other parameters. An interesting question which we did not pursue here is to what degree keeping the different parameters fixed constrains the remaining parameters given observed simulation outputs.

Other methods such as Approximate Bayesian Computation (ABC) and Simulation-based Inference (SBI) can also be used to estimate posterior distributions over parameters given specific observed model outputs, typically using statistical models based on neural networks. These methods typically iteratively sample the simulator around areas of high posterior density. One difference between this and conventional metamodelling is that for metamodelling, you typically want an accurate model over a larger domain. The metamodels can be rapidly sampled, and can in principle be plugged in to SBI and ABC frameworks, as an alternative to MCMC sampling the posterior distribution. We have not attempted to systematically compare the posterior distributions estimated from ABC or SBI to the ones estimated with MCMC from the present metamodels.

Generalization to more complex network models

The two-population model is fairly simple, and is a useful starting point since the dynamics are well known, and the simulations are cheap to run. A natural question is how well these metamodelling techniques generalize to more complex networks such as a multi-population model of cortical circuitry [51], a multi-area model of the visual system [52, 53], or a large-scale model of mouse V1 [54]. Models with more populations will have more potential model outputs to capture, and richer dynamics, and a larger number of parameters will yield a higher dimensional input space. The expressiveness of the metamodels may have to increase, and the number of samples needed to train a metamodel will likely be much higher. Of course, the number of parameters one chooses to include is arbitrary, and if one is only interested in a subset of the parameters, that can be modelled more cheaply. It is interesting to note, however, that in the present case, the power spectra mostly in the asynchronous irregular (AI) regime can be accurately captured by a low-dimensional latent representation, with only 14 dimensions used in the GPR metamodel.

In the present case, we model the external spiking input as a constant rate Poisson process, parameterised by the variable η. In more realistic networks, the input to the network will generally be more patterned. The parameters in the network we used were also kept constant, while in more realistic networks the parameters might follow a statistical distribution instead. Similar methods can be used to find the parameters of such distributions instead of the constant value as in the present case. Patterned input could also be represented in a metamodel, either through some continuous input to the model if the input changes in a continuous manner, or possibly discretely, in which case it may be necessary to train multiple metamodels for each input class separately.

The metamodels can easily be made more flexible, for the GPR model by increasing the number of GPs in either layer, and increasing the dimensionality in the latent space, and for the HCPLSR, increasing the number of clusters and/or the number of latent factors. This will lead to higher computational costs, but more complex neural networks are of course also more costly to run, so the relative benefit may not change. In the present case, sampling was done using Latin Hypercube sampling, which ensures that each parameter is uniformly sampled. This can potentially lead to a too dense sampling

in regions of the parameter space where the metamodel performs well, and too sparse sampling in regions where the metamodel performs poorly. Methods for adaptively sampling only or mostly in regions where additional sampling is needed would be an interesting direction for future research.

Use of metamodels

Once a metamodel has been trained, it can be evaluated much more rapidly than the model it is trained on. If it is well calibrated, it should be able to interpolate accurately inside the volume of the parameter space which was used to train it, and give a continuous approximation to the model behaviour. If a probabilistic metamodel is used, it may also provide closed-form approximations to otherwise intractable distributions over simulation outputs, which can in turn be used for model inversion over the domain on which the metamodel is trained. The utility of metamodels will depend on what one is interested in. For models which are computationally demanding, the benefit of a metamodel that is significantly faster will be larger than for models that are not as demanding. Trained metamodels could be distributed for use by others without requiring large computational resources. The use of metamodels for inverse modelling and investigating interactions between parameters can be useful for any model in which one is interested in the inverse model over a larger domain.

Conclusion

We have shown that for the present situation, the power spectra of the population spiking activities and the local field potential can be accurately modelled by the GPR and HCPLSR metamodels, over a domain largely consisting of activity in the AI regime. The GPR metamodel performed better than the HCPLSR metamodel, but at the cost of higher computational demands. A trained GPR metamodel can be used to estimate posterior distributions over simulation parameters given observed simulation outputs. Using the estimated posterior distributions, the population spiking activities and the LFP were found to contain similar information about most of the model parameters, apart from the synaptic strength, about which the LFP was found to contain significantly more information.

References

1. Einevoll GT, Destexhe A, Diesmann M, Grün S, Jirsa V, de Kamps M, Migliore M, Ness TV, Plesser HE, Schürmann F. The Scientific Case for Brain Simulations Neuron. 2019;4:735–744

2. Dayan P, Abbott LF. Theoretical Neuroscience. MIT Press, Cambridge; 2001.

3. Koch C. Biophysics of Computation. Oxford Univ Press, Oxford; 1999.

4. Sterratt D, Graham B, Gillies A, Willshaw D. Principles of Computational Modelling in Neuroscience. Cambridge University Press; 2011.

5. Brunel N. Dynamics of sparsely connected networks of excitatory and inhibitory neurons. Computational Neuroscience. 2000;8:183–208. doi:10.1016/S0928-4257(00)01084-6

6. Einevoll GT, Kayser C, Logothetis NK, Panzeri S. Modelling and analysis of local field potentials for studying the function of cortical circuits. Nat Rev Neurosci. 2013 Nov;14(11):770-85. doi: 10.1038/nrn3599

7. Martens H, Tøndel K, Tafintseva V, Kohler A, Plahte E, Vik JO, Gjuvsland A, Omholt S. PLS-Based Multivariate Metamodeling of Dynamic Systems Springer Proceedings in Mathematics and Statistics. 2013;56 doi: 10.1007/978-1-4614-8283-3_1

8. Tøndel K, Martens H. Analyzing complex mathematical model behavior by partial least squares regression-based multivariate metamodeling. WIREs Computational Statistics. 2014;6:440–475

9. Sunnåker M, Busetto AG, Numminen E, Corander J, Foll M, Dessimoz C. Approximate Bayesian Computation PLOS Computational Biology. 2013;9:1–10 doi:10.1371/journal.pcbi.1002803

10. Tavaré S, Balding DJ, Griffiths RC, Donnelly P. Inferring coalescence times from DNA sequence data. Genetics. 1997;145:505–518

11. Papamakarios G, Murray I Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation NIPS. 2016. 1036–1044

12. Lueckmann JM, Gonçalves PJ, Bassetto G,Öcal K, Nonnenmacher M, Macke JH Flexible Statistical Inference for Mechanistic Models of Neural Dynamics. Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017:1289-1299

13. Papamakarios G, Sterratt DC, Murray I Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows. The 22nd International Conference on Artificial Intelligence and Statistics. 2018:837-848

14. Gerstner W, Kistler, WM, Naud R, Paninski L. Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition. Cambridge University Press. 2014. doi:10.1017/CBO9781107447615

15. Schwalger T, Deger M, Gerstner W. Towards a theory of cortical columns: From spiking neurons to interacting neural populations of finite size. PLOS Computational Biology. 2017;13. doi:10.1371/journal.pcbi.1005507

16. Buzsáki G, Anastassiou C, Koch C. The origin of extracellular fields and currents–EEG, ECoG, LFP and spikes. Nature Reviews Neuroscience. 2012;13(6):407–420. doi:10.1038/nrn3241.

17. Pesaran B, Vinck M, Einevoll GT, Sirota A, Fries P, Siegel M, et al. Investigating large-scale brain dynamics using field potential recordings: analysis and interpretation. Nature Neuroscience. 2018;21(7):903–919. doi:10.1038/s41593-018-0171-8.

18. Tøndel K, Indahl UG, Gjuvsland AB et al. Hierarchical Cluster-based Partial Least Squares Regression (HC-PLSR) is an efficient tool for metamodelling of nonlinear dynamic models. BMC Syst Biol 5, 90 (2011). doi: https://doi.org/10.1186/1752-0509-5-90

19. Wold S, Martens H, Wold H. The multivariate calibration problem in chemistry solved by the PLS method. Matrix Pencils. Lecture Notes in Mathematics, vol 973. Springer. 1983. doi: 10.1007/BFb0062108

20. Wold S, Sjöström M, Eriksson L. PLS-regression: a basic tool of chemometrics. Chemometrics and Intelligent Laboratory Systems. 2001;58:109–130

21. Tøndel K, Vik JO, Martens H, Indahl UG, Smith N, Omholt SW. Hierarchical multivariate regression-based sensitivity analysis reveals complex parameter interaction patterns in dynamic models, Chemometrics and Intelligent Laboratory Systems. 2013;120:25–41

22. Rasmussen CE, Williams CKI. Gaussian Processes for Machine Learning MIT Press. 2006.

23. Hensman J, Fusi N, Lawrence ND. Gaussian Processes for Big Data. In Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence. 2013. 282–290

24. Titsias M. Variational Learning of Inducing Variables in Sparse Gaussian Processes Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics. 2009;5:567–574

25. Damianou A, Lawrence ND. Deep Gaussian Processes. Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics. 2013;31:207–215

26. Salimbeni H, Deisenroth M Doubly Stochastic Variational Inference for Deep Gaussian Processes. Advances in Neural Information Processing Systems. 2017;30:4591–4602

27. Bui T, Hernandez-Lobato D, Hernandez-Lobato J, Li Y, Turner R. Deep Gaussian Processes for Regression using Approximate Expectation Propagation. Proceedings of The 33rd International Conference on Machine Learning. 2016;48:1472–1481

28. McKay MD, Beckman RJ, Conover WJ. Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. 1979;21:239-245 doi:10.1080/00401706.1979.10489755

29. Fardet T, Vennemo SB, Mitchell J, Mørk H, Graber S, Hahne J, Spreizer S, Deepu R, Trensch G, Weidel P, Jordan J, Eppler JM, Terhorst D, Morrison A, Linssen C, Antonietti A, Dai K, Serenko A, Cai B, Kubaj P, Gutzen R, Jiang H,

Kitayama I, Jürgens B, Plesser HE. NEST 2.20.0. Zenodo. 2020. doi: 10.5281/zenodo.3605514

30. Gewaltig M-O, Diesmann M. NEST (NEural Simulation Tool) Scholarpedia. 2007.

31. Skaar J-EW, Stasik AJ, Hagen E, Ness TV, Einevoll GT. Estimation of neural network model parameters from local field potentials (LFPs) PLOS Computational Biology. 2020;16 doi:10.1371/journal.pcbi.1007725

32. Welch P. The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Transactions on audio and electroacoustics. 1967;15(2):70–73.

33. Jones E, Oliphant T, Peterson P, et al.. SciPy: Open source scientific tools for Python; 2001–. Available from: `http://www.scipy.org/`.

34. Nordlie E, Gewaltig MO, Plesser HE. Towards Reproducible Descriptions of Neuronal Network Models. PLoS Computational Biology. 2009;5(8):e1000456. doi:10.1371/journal.pcbi.1000456.

35. Hagen E, Dahmen D, Stavrinou ML, Lindén H, Tetzlaff T, van Albada SJ, et al. Hybrid Scheme for Modeling Local Field Potentials from Point-Neuron Networks. Cerebral Cortex. 2016;26(12):4461–4496. doi:10.1093/cercor/bhw237.

36. Snelson E, Ghahramani Z. Sparse Gaussian Processes using Pseudo-inputs. Advances in Neural Information Processing Systems, MIT Press. 2006:1257–1264.

37. Gardner JR, Pleiss G, Bindel D, Weinberger KQ, Wilson AG. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. Advances in Neural Information Processing Systems. 2018. 7587–7597

38. Leibfried F, Dutordoir V, John S, Durrande N. A Tutorial on Sparse Gaussian Processes and Variational Inference. arXiv:2012.13962

39. van der Wilk M, Dutordoir V, John ST, Artemev A, Adam V, Hensman J. A Framework for Interdomain and Multioutput Gaussian Processes. arXiv. 2020 doi: 10.48550/ARXIV.2003.01115

40. Haario H, Saksman E, Tamminen J. An Adaptive Metropolis Algorithm. Bernoulli, 7(2), 223–242. https://doi.org/10.2307/3318737

41. Gelman A, Rubin DB Inference from Iterative Simulation Using Multiple Sequences. Statistical Science. 1992;7:457–472.

42. Wegelin JA. A Survey of Partial Least Squares (PLS) Methods, with Emphasis on the Two-Block Case. Technical report, Department of Statistics, University of Washington, Seattle, 2000.

43. Bezdek J. Pattern Recognition With Fuzzy Objective Function Algorithms. Kluwer Academic Publishers. 1981. doi: 10.1007/978-1-4757-0450-1.

44. Berget I, Mevik BH, Næs T. New modifications and applications of fuzzy C-means methodology. Computational Statistics & Data Analysis. 52;2403–2418. 2008. doi: 10.1016/j.csda.2007.10.020

45. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.

46. Hines ML, Davison AP, Muller E. NEURON and Python. Frontiers in Neuroinformatics. 2009;3(January):1. doi:10.3389/neuro.11.001.2009.

47. Tøndel K, Indahl UG, Gjuvsland AB et al. Multi-way metamodelling facilitates insight into the complex input-output maps of nonlinear dynamic models. BMC Syst Biol 6, 88 (2012). doi: https://doi.org/10.1186/1752-0509-6-88.

48. Łęski S, Lindén H, Tetzlaff T, Pettersen KH, Einevoll GT. Frequency Dependence of Signal Power and Spatial Reach of the Local Field Potential. PLoS Computational Biology. 2013;9(7):e1003137. doi:10.1371/journal.pcbi.1003137.

49. Lindén H, Pettersen KH, Einevoll GT. Intrinsic dendritic filtering gives low-pass power spectra of local field potentials. Journal of computational neuroscience. 2010;29(3):423–444. doi:10.1007/s10827-010-0245-4.

50. Gelman A, Carlin JB, Stern HS, Rubin DB. Bayesian Data Analysis. Chapman and Hall/CRC. 2004.

51. Potjans TC, Diesmann M. The Cell-Type Specific Cortical Microcircuit: Relating Structure and Activity in a Full-Scale Spiking Network Model. Cerebral Cortex. 2014;24(3):785–806. doi:10.1093/cercor/bhs358.

52. Schmidt M, Bakker R, Hilgetag CC, Diesmann M, van Albada SJ. Multi-scale account of the network structure of macaque visual cortex. Brain Structure and Function. 2018;223:1409–1435

53. Schmidt M, Bakker R, Shen K, Bezgin G, Diesmann M, van Albada SJ. A multi-scale layer-resolved spiking network model of resting-state dynamics in macaque visual cortical areas. (2018) PLoS Comput Biol 14(10): e1006359. https://doi.org/10.1371/journal.pcbi.1006359

54. Billeh YN, Cai B, Gratiy SL, Dai K, Iyer R, Gouwens NW, Abbasi-Asl R, Jia X, Siegle JH, Olsen SR, Koch C, Mihalas S, Arkhipov A. Systematic Integration of Structural and Functional Data into Multi-scale Models of Mouse Primary Visual Cortex. Neuron. 2020;3:388–403