Metamodelling of a two-population spiking neural network

Jan-Eirik W. Skaar[1], Nicolai Haug[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, Masked Autoregressive Flows and Deep Gaussian Process Regression, to estimate the power spectra of two different signals; 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 metamodels, we invert the network, 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 the metamodels, and demonstrate that it can reliably find the different combinations of parameters that can give rise to an observed simulation output. We compare the metamodels on both the forward and inverse problem, 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 may be 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 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 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]. SNPE directly models the posterior, typically as a normalizing flow parameterized by a neural network, while SNLE models the likelihood, and infers the posterior in a separate step. The present work focuses on creating an approximation to the likelihood, as in the first step in SNLE, and can thus also be used to infer posterior distributions in the same manner. These methods can also be 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. This approach can be viewed as a metamodel that is trained locally around a specific point of interest in the parameter space. In this case, the metamodel will not have to be as flexible as in the case where one is interested in a much larger volume of the parameter space.

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 approximate the distribution over 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 metamodels approximate the conditional probability distribution $p(\mathbf{x}|\boldsymbol{\theta})$, where $\mathbf{x}$ is the power spectrum of either the population spiking activity or the LFP, and $\boldsymbol{\theta}$ is the biological parameters of the spiking network. The LFP, 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. Figure 1 shows an overview of the modelling scheme we employ.

We train metamodels based on Gaussian Process Regression (GPR) [23], and compare them to a Masked Autoregressive Flow (MAF) [18], a model often used for density estimation tasks. Both models can be trained to approximate the conditional probability densities of the simulator. Due to advances in sparse approximations to Gaussian Process (GP) models, allowing them to be used on much larger data sets [24, 25], and extensions to deep models (DGPRs) [26–28], which gives them increased flexibility, GPs provide a powerful class of nonparametric regression and classification models. MAF is a normalizing flow, a parametric model based on making successive invertible transformations, typically based on neural networks, to a simple base distribution, typically assumed to be Gaussian. It can be trained either as a conditional distribution, where the neural networks take an additional input representing the condition, or as a simpler distribution without any conditioning. Models representing conditional distributions are particularly useful as a metamodel for systems with stochastic dynamics, since that implies that there is a distribution over possible outputs that has to be modelled. We additionally train a Hierarchical Cluster-based Partial Least Squares Regression [19](HCPLSR), a model that has been used previously for metamodelling of biological systems but only gives a point-estimate of the simulator output [8, 22].

We show that the simulator can be accurately captured by the metamodels, and that the DGPR metamodel is a competetive model for estimating the complex function described by the network model, and that accurate forward models can be trained. We estimate the posterior distributions separately for DGPR metamodels trained on the population spiking activities and the LFP, for a variety of different simulation outputs.

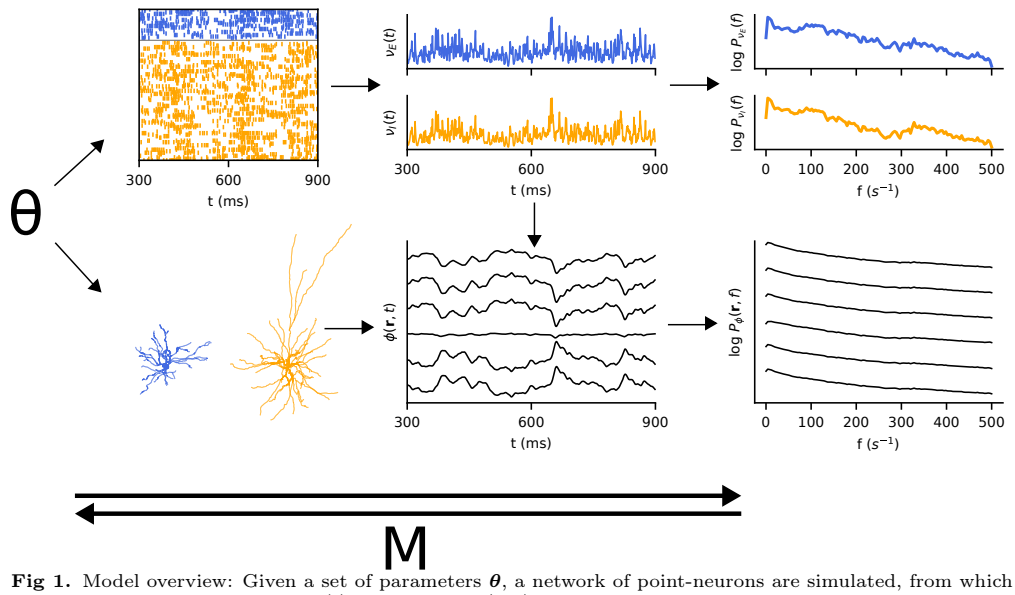

Fig 1. Model overview: Given a set of parameters $\boldsymbol{\theta}$, a network of point-neurons are simulated, from which the population spiking activities $\nu(t)$ are computed (top). With morphological neurons spatially extended in a column, the synaptic spiking activities from the point-neuron 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.

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 its membrane potential reaches a firing threshold V_{thr}, the neuron 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 t_{ref}. 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_{s}. 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 $\eta = \frac{\nu_{\mathrm{ext}}}{\nu_{\mathrm{thr}}}$, where $\nu_{\mathrm{thr}} = V_{\mathrm{thr}} C_m / (J \tau_m e \tau_s)$, 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 [29], 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. 10000 simulations were run using the NEST simulator [30, 33], comprising both the training data set and the test data set. The regions of the parameter space giving strongly synchronous firing, discussed in [5, 34], 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 distributions (see Sections 2.4 and 3.4). 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 $[\psi](f)^*[\psi](f)/T$ for some signal $\psi(t)$, where $[\psi](f)$ denotes the Fourier transform of ψ, and T is the window length, was estimated using Welch's method [35] for both the population spiking activities and the LFP. The implementation in the Python SciPy [36] package

(`scipy.signal.welch`) was used, with a Hann window of length 256 and an overlap of 128. The metamodels are trained on the base 10 logarithm of these power spectra.

Table 1. Description of point-neuron network following the guidelines of [37].

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 alpha-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 [38] (github.com/INM-6/hybridLFPy), which allows for the decoupling of the simulation of spiking dynamics (here computed using point

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	$[100 - 300]$ pF
t_{d}	synaptic delay period	$[0.1 - 3.0]$ ms
t_{ref}	absolute refractory period	$[0.1 - 4.0]$ ms
τ_s	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

neurons) and predictions of extracellularly recorded LFPs. The latter part relies on 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 [38], 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 alpha-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	$[15, 30]$ ms
c_{m}	membrane capacitance	$[0.6, 1.4]$ 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 [34, 38], 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 [34, 38] 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, and hence synaptic parameters. In order to account for the membrane parameters τ_m and c_m, where c_m is the per-area capacitance, we compute the kernel for a single time step on a grid of different values of τ_m and C_m, and adjust this kernel by linear interpolation. We compute the kernels at the values $[0.4, 0.6, 0.8, 1.0, 1.2]$ for c_m and $[15, 18, 21, 24, 27, 30]$ for τ_m. Since the membrane capacitance for morphological neurons is dependent on the area of the neuron, we choose a baseline value of $c_m = 1.0$, and adjust it by the same proportion as C_m for the

point-neuron network. The synaptic parameters can subsequently be accounted for in a separate step.

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* $m(\boldsymbol{\theta})$ and a *covariance function* $k(\boldsymbol{\theta}, \boldsymbol{\theta})$ [23]. Consider a data set consisting of n inputs of dimension d contained in the matrix Θ, and corresponding outputs $\mathbf{x}$, which are assumed to be one-dimensional for notational simplicity in this section. A Gaussian process regression model assumes a distribution over an underlying latent function $f(\boldsymbol{\theta})$ which is modelled as a Gaussian process. Evaluating this distribution at the set of input points Θ gives a joint Gaussian distribution $f(\Theta) \sim \mathcal{N}(\mathbf{0}, k(\Theta, \Theta))$, where the covariance function $k(\Theta, \Theta)$ must return a positive semi-definite covariance matrix between all the data points contained in Θ. 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. In the present work, we use the Matérn 5/2 kernel

$$C_{5/2}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma^2\big(1 + 5d + \frac{5}{3}d^2\big)\exp(-5d), \;\; d = \sqrt{5\sum_i \left(\frac{\theta_i - \theta_i'}{l_i}\right)^2}, \tag{11}$$

where $\mathbf{l}$ and σ are learnable parameters.

The model of the observed outputs $\mathbf{x}$ is given by the underlying latent function $\mathbf{f}(\Theta)$ corrupted by some noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 I)$ such that $p(\mathbf{x}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$. The marginal likelihood of the data is obtained as

$$p(\mathbf{x}; \Theta) = \int p(\mathbf{x}|\mathbf{f}; \Theta) p(\mathbf{f}; \Theta) d\mathbf{f} = \mathcal{N}(\mathbf{0}, (K(\Theta, \Theta) + \sigma_n I)^{-1}), \tag{12}$$

which can be found analytically in the case of a Gaussian likelihood $p(\mathbf{x}|\mathbf{f}, \Theta)$, and is

used to optimize the model parameters.

Given a new set of inputs Θ_* on which we would like to evaluate the model, the GP specifies a joint distribution over $\mathbf{x}$ and $\mathbf{x}_*$, and the predictions are made as the distribution over $\mathbf{x}_*$ conditioned on the training data $\mathbf{x}$ [23]

$$p(\mathbf{x}_*|\mathbf{x};\Theta_*,\Theta) = \int p(\mathbf{x}_*|\mathbf{f}_*;\Theta_*)p(\mathbf{f}_*|\mathbf{f};\Theta,\Theta_*)p(\mathbf{f}|\boldsymbol{\theta};\Theta)d\mathbf{f}d\mathbf{f}_* \tag{13}$$

$$= \mathcal{N}(\mathbf{x}_*|K_{n*}K_{nn}^{-1}\mathbf{x}, \quad K_{**} - K_{f*}^T(K_{nn}+\sigma_n^2 I)^{-1}K_{f*} + \sigma_n^2 I), \tag{14}$$

where we use the shorthand notation $K_{nn} = k(\Theta,\Theta)$, $K_{n*} = k(\Theta,\Theta_*)$ and $K_{**} = k(\Theta_*,\Theta_*)$. The semicolon indicates the input points corresponding to the distributions, on which the covariance function is evaluated. We can immediately see that the mean value of the distribution at the predicted locations Θ_* are linear combinations of the training values $\mathbf{x}$.

Deep, or composite, GP models are constructed by iteratively defining GPs with the previous GP as inputs. A joint prior distribution over all layers is given by

$$p(\mathbf{x},\{\mathbf{f}^l,\}_{l=1}^L) = \prod_{i=1}^N p(x_i|f_i^L)\prod_{l=1}^L p(\mathbf{f}^l;\mathbf{f}^{l-1}) \ , \tag{15}$$

where $\mathbf{f}^0 = \boldsymbol{\theta}$. As in standard GP models, we would like to marginalize out the latent functions in all layers for optimization, and for condition them on the training data and then marginalize them for prediciton. This is obviously not possible analytically, and approximations have to be used. We do not go in to details, but refer the readers to [41] for an excellent tutorial.

For multi-dimensional outputs, one can either assume the outputs to be independent, or correlated. In the former case, a separate GP is indepedently trained for each output dimension. For the latter case, one simple way of introducing correlations between output channels is to make a linear combination of independent GPs. If the number of independent GPs is lower than the number of output dimensions, a low-dimensional representation of the output is achieved, with a linear mapping between the latent space and the output. The model can be written as

$$\mathbf{g}(\boldsymbol{\theta}) = \{g_p(\boldsymbol{\theta})\}_{p=1}^P, \qquad \mathbf{f}(\boldsymbol{\theta}) = W\mathbf{g}(\boldsymbol{\theta}), \tag{16}$$

where $g_p(\boldsymbol{\theta})$ forms a set of independent GPs, and W is an $n \times p$ matrix used to linearly construct the output $\mathbf{f}(\boldsymbol{\theta})$ [42].

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 DGPR model based on the framework by [27]. We use 2 layers, and do a parameter scan in order to find the number of GPs to use in each layer. The final output is constructed by a linear transformation from the GPs in the second layer as given by Eq. 16.

2.4 Posterior distribution over parameters

The metamodels provide 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 [43]. 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 and considered the burn-in period. 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 [44] was computed independently for each parameter, for each distribution, in order to assess the convergence properties of the chains.

2.5 Technical details

Reproducibility

The simulations were run using Python v3.10.5. All point-network simulations were run with the NEST simulator v3.3 [30, 33]. The forward-modeling of the LFP was done using hybridLFPy v0.2 [38] with LFPy v2.3 [31]. The DGPR metamodels were trained using GPyTorch v1.9.1 [40] with PyTorch v1.13.1. The MAF metamodels were trained using SBI v0.21.0 [32]. The metamodels were trained on NVIDIA Quadro RTX 8000 GPUs. All simulation and analysis code for this study is available on Github: https://github.com/janskaar/Skaar_bioRxiv_506616v1

3 Results

The aim of this study was to apply metamodelling techniques to simulations of neuronal networks. Specifically, we applied Deep Gaussian Process Regression (DGPR) and Masked Autoregressive Flow (MAF) models to approximate the likelihood of the network simulator model. In all cases, we consider the output of the simulator to be the power spectrum of either the summed population spiking activity, or the local field potential generated by the network. We compare the accuracies of the forward predictions of all three models. For the DGPR and MAF metamodels, we also estimate the posterior distribution for 100 examples in the test data set, and evaluate the accuracy of these based on the posterior predictive distribution.

We start by showing some example simulation outputs, and discuss the accuracy of the forward predictions of the metamodels. We then evaluate the number of training samples needed for the metamodels to obtain high prediction accuracy, before we discuss the inverse modelling and the posterior distributions. A HCPLSR metamodel was also trained on the same task. We refer to Supplementary section 3 for the results from that model.

3.1 Simulations

A total of 10000 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

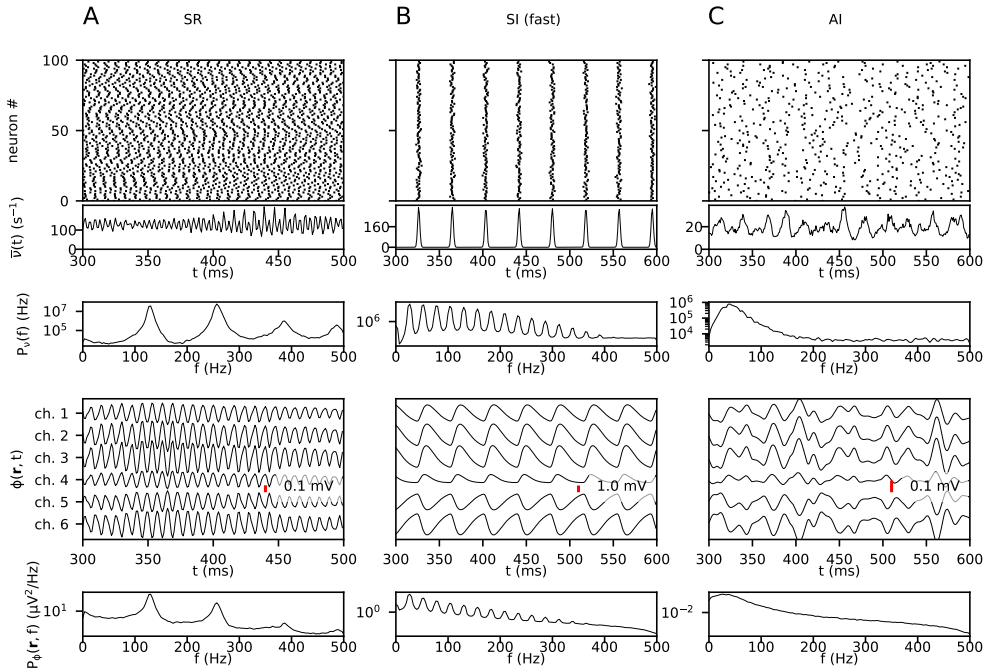

Fig 2. Example activity states. Each column shows, from top to bottom, a raster plot, population firing rates, the power spectrum of the population averaged firing rates, the LFP, and the power spectrum of the LFP. The raster plot and firing rates are taken from the excitatory population. A: Synchronous regular activity. B: Synchronous irregular activity. C: Asynchronous irregular activity.

variety of activity states, characterized by whether the neurons fire in *synchrony*, and how *regularly* they fire, that is, 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 power spectrum. Figure 2 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.

3.2 Forward predictions

The metamodels were tested on a data set containing 1681 examples, jointly sampled with the training data using Latin Hypercube sampling. Models were trained on data sets ranging from 100 to 8000 examples in order to examine the density of samples needed to adequately represent the simulation output space. A grid search of hyperparameters were was for each metamodel in order to find a suitable model for the problem at hand, which can be found in Supplementary section 1. The metamodel obtained from the hyperparameter search were used for all subsequent experiments, and where we trained new metamodels, the hyperparmeters were the kept the same. Figure 3A and B show examples of simulated outputs and corresponding outputs from the metamodels for two different sets of parameter values, for the population spiking activities and LFPs respectively. Only the outputs 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 the colored lines show the output by the metamodels. The shaded area shows two standard deviations above and below the mean for the DGPR and MAF metamodels. Visually inspecting the top example in panel A, both metamodels accurately capture the mean of the simulations for all frequencies, but slightly overestimate the variance, particularly for the lower frequencies. In the bottom example, there is a larger discrepancy between the MAF metamodel and the simulator for middle, while the DGPR metamodel is accurately able to capture both the mean and variance. Note that the covariance between frequency channels is not considered at all in this figure. In panel B, the LFP for the same simulations and the DGPR

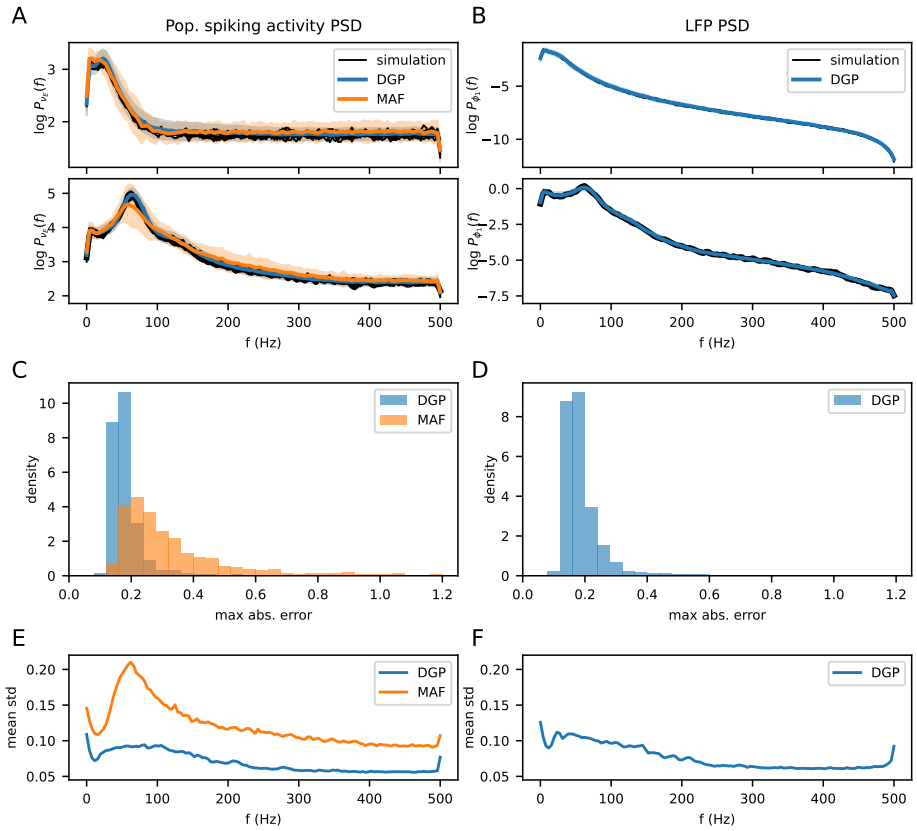

Fig 3. A: For two different parameter sets, the power spectrum of the population spiking activity from 10 simulations (black lines), and metamodels (blue and orange lines) are shown. The shaded area shows two standard deviations of the distribution given by the metamodels. B: Same as A, but for the LFP. Only the DGPR metamodel was trained on the LFP. C: Distribution over maximum (over frequencies) absolute errors for both metamodels, on the power spectrum of the population spiking activity. D: Same as for C, but for the LFP. E: Mean standard deviation of the metamodel outputs, evaluated at the parameters in the test data set. F: Same as E, but for the LFP.

prediction are shown. Generally, the LFPs are smoother and the variance is lower compared to the population spiking activity. Figure 3C and D shows the distribution over maximum (over frequencies) absolute errors on the test data set, where the expectation of the metamodel is used as the prediction value. We show 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, rather than small deviations, which are expected to occur. The expectation of distribution given by the DGPR metamodel more closesly matches the simulated values compared to the MAF metamodel. Figure 3 E and F shows the mean standard deviation of the metamodel distributions evaluated at the test data parameters. The DGPR metamodel produces narrower distributions over the simulator output compared to the MAF model.

In terms of computational cost, evaluating the log probability of 50 example outputs takes around 4 seconds with the DGPR metamodel, and 0.014 seconds with the MAF metamodel. Generating 50 samples from the metamodels takes 1.6 seconds for the MAF metamodel, and 0.07 seconds with the DGPR metamodel.

3.3 Number of training samples needed to represent the model

The DGPR metamodel was trained on data sets ranging from 100 to 8000 examples of the population spiking activities, and the performances on the same test data set were evaluated. Figure 4 shows the maximum (over frequencies) absolute prediction errors, 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 to generalize better. The prediction errors on the training data set increases, as it becomes more heterogeneous and the model is less able to overfit to it. 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, indicating that the parameter space is sufficiently densely sampled.

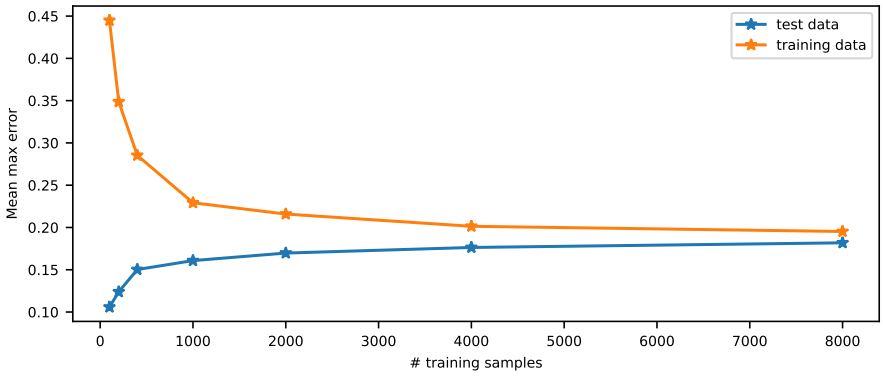

Fig 4. Mean (over test data set and training data set) of the maximum (over frequencies) errors, in absolute scale, as a function of the number of examples used to train the metamodels on the population spiking activities.

3.4 Inverse modelling

Using the conditional distribution over simulation outputs given parameter values, estimated by the metamodels, posterior distributions for 100 simulations in the test data set were estimated. Separate posterior distributions for both both metamodels conditioned on the population spiking activities, and for the DGPR metamodel conditioned on the LFP, were estimated using 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 each 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 in order to evaluate the accuracy of the posterior distribution.

Figure 5A 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

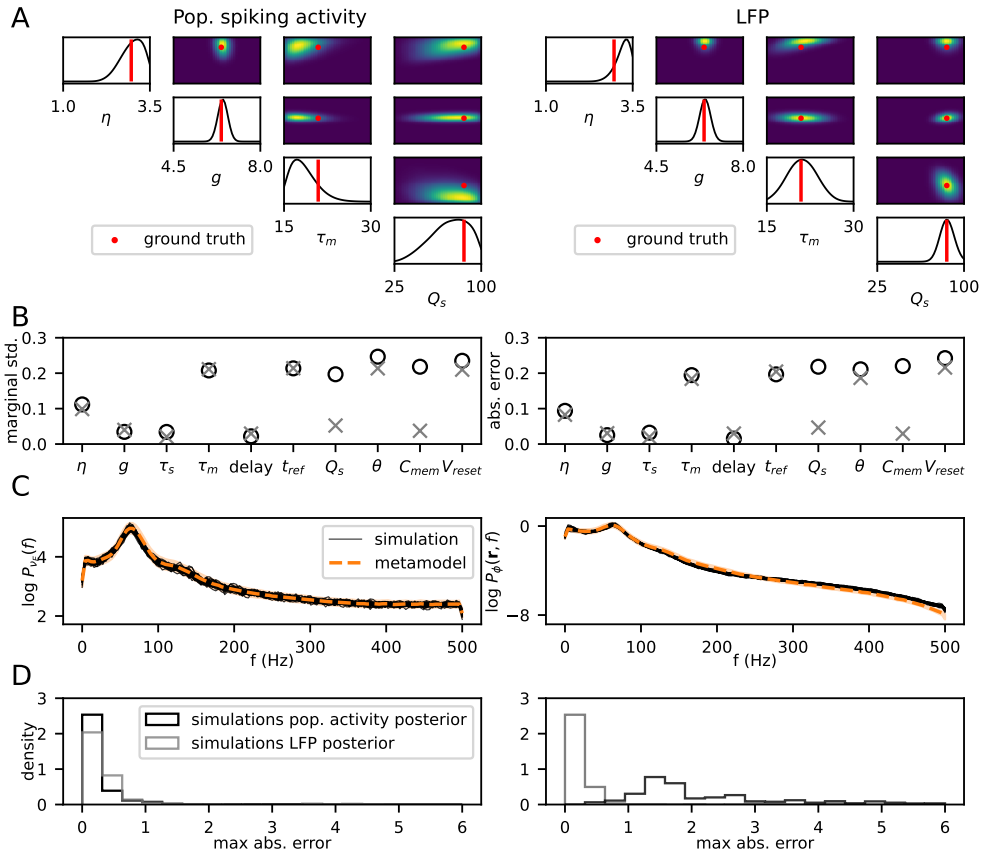

Fig 5. In A, C and D, the left-hand and right-hand side show equivalent plots for the population spiking activities and LFPs respectively. A: 1D and 2D marginal posterior distributions over a subset of the parameters, for an example in the test data set. The red dots and bars show the parameter values 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 ground-truth simulation output from which the posterior distribution was computed, and the simulation output from the posterior distributions (black lines in C). Left plot shows the errors from the population spiking activities, right plot shows the errors from the LFP. The black lines show the simulations from the posterior conditioned on the population spiking activities, the gray lines show the simulations from the posterior conditioned on the LFP.

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 time constant 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 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.

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 5B 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 5B 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 5A) 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 the posterior distribution is conditioned on.

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 , and the membrane capacitance, C_{mem}, are significantly more constrained by the LFP than by the population spiking activities, with a standard deviation more than three times as large for the population spiking activities. This can also be observed in the example distributions shown in Fig 5A. 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. The full posterior distribution from Fig 5A is shown in Supplementary section 2.

Fig 5C 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 5D shows the distribution of the maximum (taken over frequencies and simulations for a given posterior) of the simulation outputs run with parameters drawn from the posterior distributions, and the simulation output the posterior is conditioned on, for the 100 posterior distributions for which the extra simulations were run.

The black line shows the errors 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, while the right-hand side shows the distances for the LFP. For both the population spiking activities and the LFP, the simulations from the posterior distributions are very similar to the signal they were conditioned on. Simulations from the posterior distributions conditiond on the LFP, also give population spiking activities very similar to the ground-truth. Simulations from the posterior distributions conditioned on the population spiking activities, on the other hand, do not give as similar LFPs to the ground-truth, which again suggests that the LFP constrains the parameters to a greater degree than the population spiking activities.

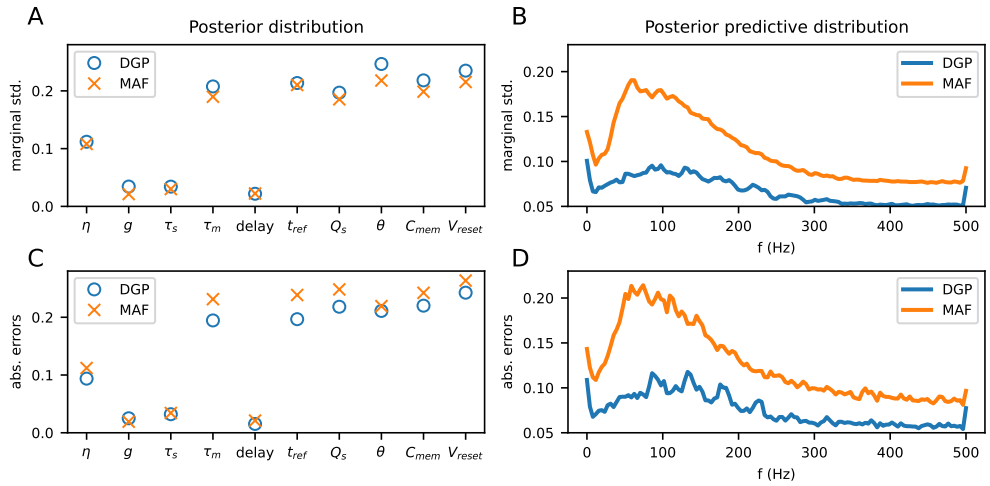

Fig 6. A: mean marginal standard deviations of the posterior distributions. B: mean marginal standard deviations of the posterior predictive distributions. C: mean distance between expectation of marginal posterior distribution and the parameters from the simulation the posterior distribution is conditioned on. D: Same as C, but for the posterior predictive distribution.

Figure 6 shows a comparison between the posteriors obtained by the DGPR metamodel and the MAF metamodel. Panel A shows the marginal standard deviations, averaged over all posterior distributions. For both metamodels, they are similar, but the DGPR is generally slightly wider. Panel B shows the same plot, but for the posterior predictive distribution, i.e. the distribution over outputs when parameters are sampled from the posterior distribution. The posterior predictive distributions from the DGPR metamodel are significantly narrower than those from the MAF metamodel, even with wider posterior distributions, indicating that the posterior distributions are more correct compared to those from the MAF metamodel. Panels C and D show the mean absolute distance between the expectation of the posterior distribution and posterior predictive distribution respectively, and the simulations they are conditioned on. For both models, the errors are on the same scale as the standard deviations, indicating that neither systematically misses the simulation it is conditioned on.

The Pearson correlation coefficient was computed for all pairwise combinations of parameters for all sampled posterior distributions. Fig 7 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 conditioned on the population spiking activities, and the lower triangle shows the coefficients conditioned on the LFP. The correlations between most of the parameters are fairly weak, but since

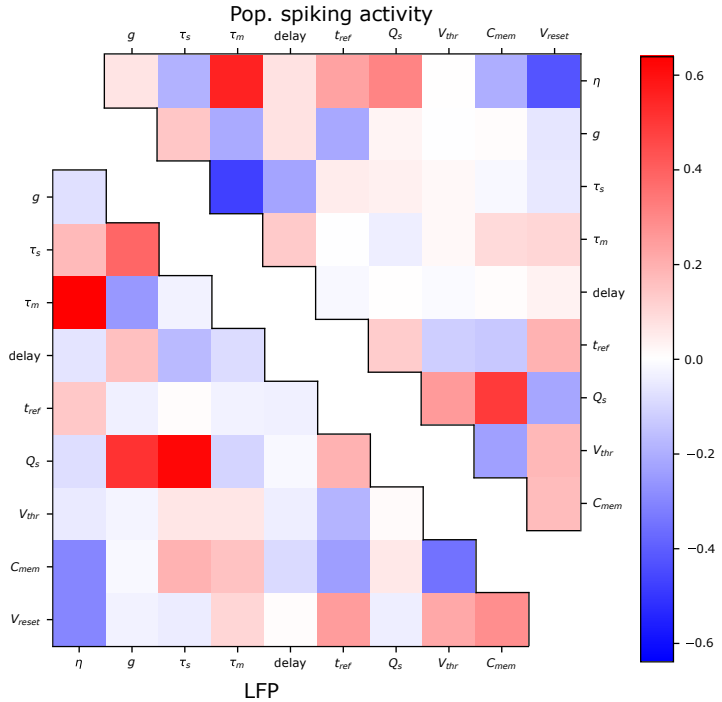

Fig 7. Pearson correlation coefficients between samples from posterior distribution over parameters, averaged over the test data set. Upper triangle shows the correlation coefficients for the posterior distributions from the simulated power spectra from the population spiking activities, while the lower triangle shows the same correlation coefficients from the LFP.

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 parameters, i.e. the synaptic strength Q_{s}, 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. Present for the distributions conditioned on both the LFP and the population spiking activities, are positive correlations are present between the parameters η and τ_{m}, as well as negative correlations between η, V_{reset} and C_{mem}. As these parameters all impact the amount of input required for a neuron to spike, there are possibly complex interactions between all these parameters yielding manifolds of similar activities and LFPs.

4 Discussion

In the present work we have trained metamodels to estimate the power spectra of population spiking activities and local field potentials (LFPs) 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 (DGPR) model, and a faster Masked Autoregressive Flow (MAF) model, and used the probabilistic nature of the metamodels to estimate posterior distributions over parameters given observed simulation outputs. DGPR models have not been much used for statistical modelling in the field of neuroscience before, and we choose to compare it the the MAF as it is a commonly used model for both forward modelling in the case of SNLE and inverse modelling in the case of SNPE.

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 small number of kernels, which can then be used to approximate the LFP for all simulations. The synaptic parameters can be fully adjusted for by a single simulation of a kernel, but the neuron membrane conductance and capacitance, cannot be adjusted for analytically. Kernels were therefore computed on a grid of these two parameters, and regression was done to approximate the true kernel for any value of the membrane conductance and capacitance. As shown in Supplementary section 4, the approximation works well, although there are some small deviations, particularly in channels 4 and 5.

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. This will also have excluded some AI-like simulations in the vicinity of the boundary between AI and SR, but we consider the difference (starting at $g = 4.5$ instead of $g = 4.0$) to be sufficiently small as to not

significantly diminish the utility of the models. 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 DGPR metamodel small. The low-rank approximation is a limiting factor for the DGPR metamodel, and ideally one would not have to limit the input domain in such a way. Increasing the dimensionality of the latent space significantly is possible and would possibly also have resulted in better accuracy, also in the case of strongly synchronous activity, 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 DGPR and MAF 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 around 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.

Theoretical approaches based on linear-response theory can provide analytical approximations to the PSDs [51–54] in the asynchronous irregular regime. The

theoretical models necessarily make assumptions about the underlying physics of the system, which also makes it possible to interpret them in ways that are not possible with purely data-driven methods. Data-driven methods are complementary to theoretical ones in the sense that they do not make any underlying assumption about the model, but comes at the price of interpretability and needing to run simulations to train the models.

The forward modelling of the population spiking activities and the LFP is similar in terms of difficulty, where the errors of the DGPR metamodels are similar. 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 [55, 56]. Comparing the DGPR metamodel to the MAF metamodel, we found that the DGPR metamodel provided better accuracy, but comes at the price of a higher cost of training. When it comes to running the trained models, the MAF is significantly faster than DGPR for evaluating the log probability, and therefore also when estimating posterior distributions. For purely sampling the forward model, the DGPR is faster than the MAF metamodel. This is not surprising, as the MAF model by design is good for evaluating log probabilities, at the cost of being slow at sampling.

The MAF metamodel suffered significantly more from overfitting compared to the DGPR metamodel. Simply increasing the number of simulations used to train the models would likely reduce the differences between the two metamodels. It is also possible that more regularization techniques could have increased the performance of the MAF, although utilizing dropout did not significantly improve the model.

For inverse modelling, the benefit of using the more computationally demanding DGPR will likely be smaller if one is doing sequential inference, and the region in which the metamodel must be accurate is smaller. Also depending on the complexity of the simulator one is approximating, in cases where one wants an accurate model over a larger domain, increased accuracy may be well worth the increased computational cost.

The HCPLSR metamodel, shown in Supplementary section 3, is significantly faster than both the DGPR and MAF metamodels, but at the cost of only giving a point-estimate of the simulation output, and less flexibility. For cases where one is only

interested in the forward modelling, such alternatives can also be an option.

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 [57]. 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 5B). Notably the distribution over most neuron parameters were wide, while the distribution over synapse and input parameters were narrower. Interestingly, the synaptic strength (Q_s) and membrane capacitance (C_m) were 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 [34] 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.

Comparing the posterior distributions generated from the DPGR metamodel to the posterior distribution generated from the MAF metamodel, we find that the the posterior distributions from the DGPR metamodel are wider, while simultaneously having a significantly narrower posterior predictive distribution. In other words, it is able to find a larger region of the parameter space that gives more similar outputs compared to the MAF metamodel. This shows the importance of accurate forward modelling in order to find good posterior distributions, an area where DGPR metamodels can be useful also in other settings when dealing with data from complex simulations.

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. Our work is closely related to SBI, particularly SNLE [13]. The approach in the present work can be considered the same as SNLE run for a single round, where we use the DGPR metamodel as a likelihood estimator. Thus, in cases where one is interested in estimating posterior distributions, particularly in cases where one has limited data and model flexibility is a concern, DGPR metamodels may be of use in the SNLE framework. Methods such as Sequential Neural Variational Inference [58] for estimating the posterior distribution instead of MCMC sampling can also readily be used with the DGPR metamodel.

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 [59], a multi-area model of the visual system [60, 61], or a large-scale model of mouse V1 [62]. 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.

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 DGPR metamodel by increasing the number of GPs in either layer, and increasing the dimensionality in the latent space, and for the MAF metamodel by increasing the number of transformations or increasing the number of hidden features in each transformation. 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.

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.

For experimental data, metamodelling may be of interest in different aspects. In the context of SBI, metamodels can be used as likelihood estimator in order to find parameters fitting network models to observed experimental data, in a similar way that has been done in this work. This would require larger and more realistic network models than the one used here, but the principle is the same. In the context of forward modelling, metamodels could be applied to experimental data to model effects of experimental parameters and their effects on experimental outcomes. How well this would work in practice, however, is not known. It would require training to specific experimental settings, and could be an interesting direction of future research.

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 DGPR and MAF metamodels, over a domain largely consisting of activity in the AI regime. The DGPR metamodel performed better than the MAF metamodel, but at the cost of higher computational demands for training and evaluation. When dealing with complex models, utilizing DGPR for forward modelling may therefore be a good alternative, particularly when the amount of data is a limiting factor, and the increased training time is not as much of an issue. The trained metamodels 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 and membrane capacitance, 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. Papamakarios G, Pavlakou T, Murray I. Masked Autoregressive Flow for Density Estimation. 2017

19. 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

20. 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

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

22. 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

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

24. 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

25. 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

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

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

28. 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

29. 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

30. 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

31. Hagen E, Ness TV LFPy. doi:10.5281/zenodo.6414521 https://github.com/LFPy/LFPy

32. Tejero-Cantero A, Boelts J, Deistler M, Lueckmann JM, Durkan C, Gonçalves PJ, Greenberg DS, Macke JH sbi: A toolkit for simulation-based inference Journal of Open Source Software. 5. 52. doi:10.21105/joss.02505

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

34. 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

35. 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.

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

37. 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.

38. 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.

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

40. 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

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

42. 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

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

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

45. 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.

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

47. 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

48. 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.

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

50. 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.

51. Lindner B, Doiron B, Longtin A Theory of oscillatory firing induced by spatially correlated noise and delayed inhibitory feedback Phys. Rev. E 6, 72 (2005) doi: 10.1103/PhysRevE.72.061919

52. Deger M, Schwalger T, Naud R, Gerstner W Fluctuations and information filtering in coupled populations of spiking neurons with adaptation Phys. Rev. E 90, 6 (2014) doi: 10.1103/PhysRevE.90.062704

53. Bos H, Diesmann M, Helias M Identifying Anatomical Origins of Coexisting Oscillations in the Cortical Microcircuit. PLoS Comput Biol 12(10): e1005132. doi: https://doi.org/10.1371/journal.pcbi.1005132

54. Trousdale J, Hu Y, Shea-Brown E, Josić K Impact of Network Structure and Cellular Response on Spike Time Correlations. PLoS Comput Biol 8(3): e1002408. doi: https://doi.org/10.1371/journal.pcbi.1002408

55. 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.

56. 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.

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

58. Glöckler M, Deistler M, Macke JH. Variational methods for simulation-based inference arXiv. 2203.04176. 2022.

59. 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.

60. 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

61. 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

62. 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

5 Supplementary material

5.1 Hyperparameter scans

For both the DGPR and MAF metamodels, a hyperparameter grid search was run in order to find suitable hyperparameters. The training procedure was the same for both models. Optimization was done using the Adam optimizer, and and the loss function was computed for the full training and test data sets every other epoch. If the loss function on the test data set had not improved by a value more than 1 in 20 epochs, training was terminated. The model was saved every time the loss function on the test data set improved.

5.1.1 DGPR

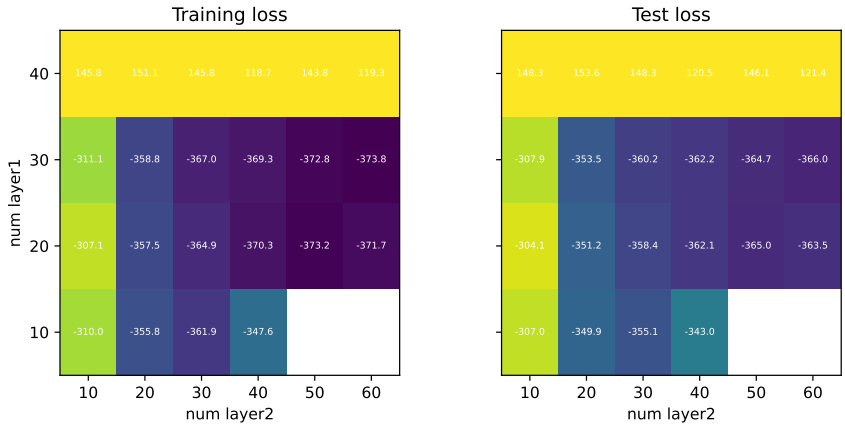

Fig 8. Loss values for different hyperparameter values of the DGPR metamodel. White squares are NaNs.

For the DGPR metamodel, the number of GPs in the first layer and the number of latent GPs in the second layer were varied. The number of layers were found based on trial and error, where we found that 1 layer performed worse, likely due to lack of expressiveness, and 3 layers or more also performed worse, likely due to increased optimization difficulties. The number of inducing points were kept at 128 for each layer, it was found that increasing it did not improve the model significantly. Each model was trained with learning rates of 0.01 and 0.001. Figure 8 shows the results for the DGPR metamodel. The best model from either of the learning rates are shown. Based on this, the model with 20 GPs in the first layer and 50 GPs in the second layer was chosen.

5.1.2 MAF

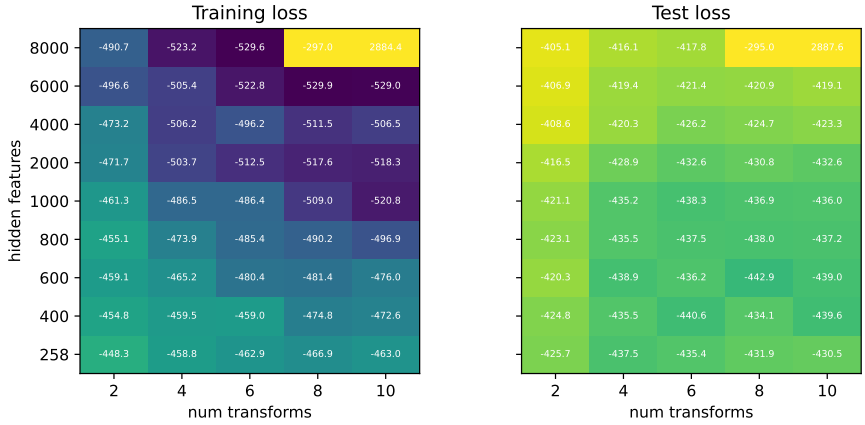

Fig 9. Loss values for different values of hyperparameters of the MAF.

For the MAF, number of transforms, the number of hidden features in the MADE block, and whether a dropoutrate of 0.5 was applied or not, were varied. Each model was trained with learning rates of 0.001 and 0.0001. Figure 9 shows the best outcome for each combination of hidden features and number of transforms.

5.2 Full posterior plots

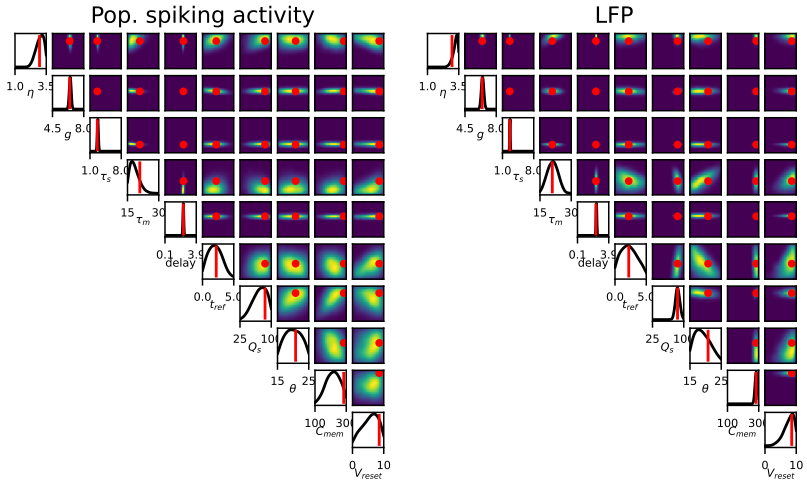

Fig 10. Full posterior from which the subset in Figure 5 is taken.

5.3 Hierarchical Partial Least Squares Regression

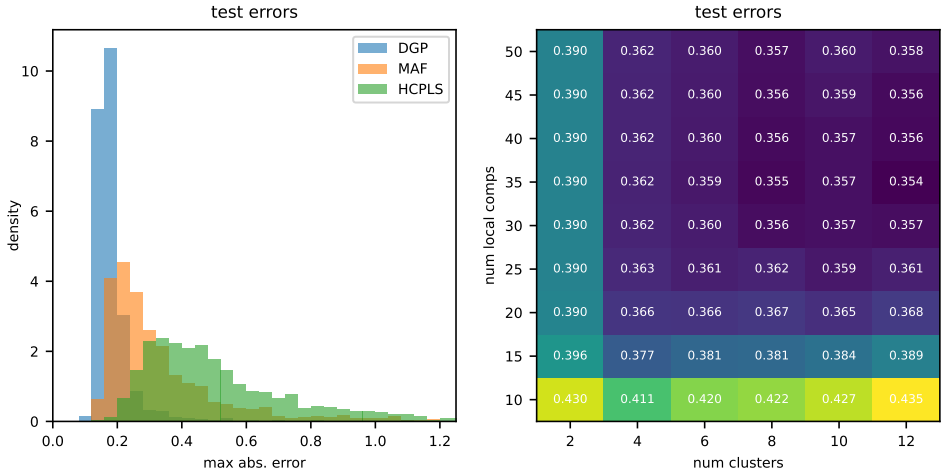

Fig 11. The left-hand side of the figure shows the distribution over maximum (over frequencies) errors on the test data set for all three metamodels. The right-hand side shows the results from the hyperparameter search for the HCPLSR metamodel, for two global components.

In addition to the DGPR and MAF metamodels, we also trained a Hierarchical Partial Least Squares Regression (HCPLSR) model [19]. HCPLSR is an extension of Partial Least Squares Regression (PLSR) [20], which has gained popularity in

chemometrics [21]. HCPLSR has also previously been used for metamodelling of different biological dynamical systems [8, 22].

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) [46, 47] 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.

We trained a HCPLSR metamodel on the same data set as the DGPR and MAF metamodels. A basis expansion of the metamodel inputs (i.e. parameters of the network model) were done to include second order terms. A hyperparameter search was done over the number of global components, number of local components and number of clusters. The left-hand side of Figure 11 shows the distribution over test errors of all three metamodels. The HCPLSR model is less accurate than both the DGPR and MAF metamodels, indicating that it is not sufficiently flexible compared to the other two metamodels. The right-hand side of Figure 11 shows the test errors achieved with different numbers of local components and the number of clusters. The best results were achieved with two global components. In terms of computational cost, the HCPLSR is significantly faster than both other metamodels, and evaluating the model at 50 locations takes only a few milliseconds.

5.4 Validation of LFP computation scheme

In order to validate the LFP computation scheme, we ran a full simulation using hybridLFPy for 3 different simulations, and compared the results. Figure 12 shows four examples in which the full hybridLFPy scheme was run and compared with our approximation scheme. The parameters were drawn from a uniform prior on the same domain as the main data set used. There are some deviations, particularly in channel 5, but the approximation is generally very good.

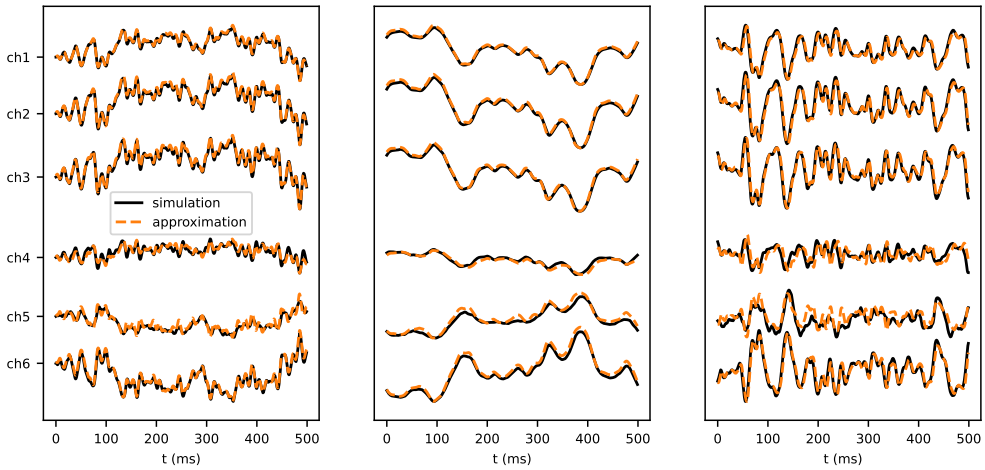

Fig 12. Three example simulations of the LFP using the full hybridLFPy scheme (black lines) and the approximation (orange dashed lines).