Proceedings of Machine Learning Research vol 145:1–28, 2021 2nd Annual Conference on Mathematical and Scientific Machine Learning

Parameter Estimation with Dense and Convolutional Neural Networks Applied to the FitzHugh–Nagumo ODE

Johann Rudi JRUDI@ANL.GOV
Julie Bessac JBESSAC@ANL.GOV
Amanda Lenzi ALENZI@ANL.GOV
Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL

Editors: Joan Bruna, Jan S Hesthaven, Lenka Zdeborova

Abstract

Machine learning algorithms have been successfully used to approximate nonlinear maps under weak assumptions on the structure and properties of the maps. We present deep neural networks using dense and convolutional layers to solve an inverse problem, where we seek to estimate parameters of a FitzHugh–Nagumo model, which consists of a nonlinear system of ordinary differential equations (ODEs). We employ the neural networks to approximate reconstruction maps for model parameter estimation from observational data, where the data comes from the solution of the ODE and takes the form of a time series representing dynamically spiking membrane potential of a biological neuron. We target this dynamical model because of the computational challenges it poses in an inference setting, namely, having a highly nonlinear and nonconvex data misfit term and permitting only weakly informative priors on parameters. These challenges cause traditional optimization to fail and alternative algorithms to exhibit large computational costs. We quantify the prediction errors of model parameters obtained from the neural networks and investigate the effects of network architectures with and without the presence of noise in observational data. We generalize our framework for neural network-based reconstruction maps to simultaneously estimate ODE parameters and parameters of autocorrelated observational noise. Our results demonstrate that deep neural networks have the potential to estimate parameters in dynamical models and stochastic processes, and they are capable of predicting parameters accurately for the FitzHugh–Nagumo model.

Keywords: Parameter estimation; Inverse problem; Reconstruction maps; Dense and convolutional neural networks; Nonlinear ordinary differential equation; FitzHugh–Nagumo.

1. Introduction

We consider inverse problems with the aim of estimating parameters from given observational data, where the parameters give rise to the data through the solution of a parametrized physical model. Such inverse problems have in the past been solved deterministically with techniques from optimization (Tarantola, 2005) or in a statistical/Bayesian framework using sampling methods, particle filters, and Kalman filters (Kaipio and Somersalo, 2005). The current work investigates new approaches to solving particular inverse problems that exhibit computational challenges prohibiting effective use of the established solution techniques mentioned above. We propose to computationally learn solution operators for inverse problems based on deep neural networks because of their well-known potential of finding generalizable nonlinear maps. We explore neural network (NN) architectures consisting of a sequence of dense, or fully connected, layers and convolutional neural networks (CNNs).

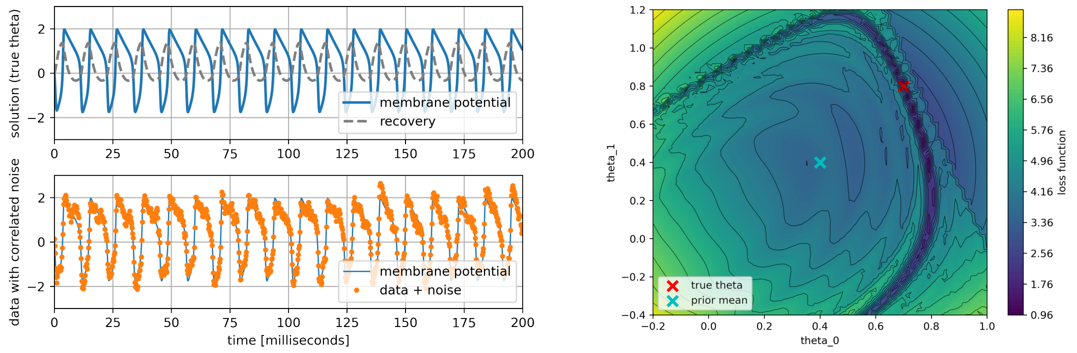

Figure 1: Left, top graph: Solution of FitzHugh–Nagumo system of two ODEs (blue and gray lines) generated with true parameters of the inverse problem. Left, bottom graph: Data (orange dots) of inverse problem stemming from the membrane potential of the solution with additive correlated noise. Right: Manifold of highly nonlinear loss function (colors and contours), with respect to parameters θ_0 and θ_1, of the inverse problem governed by the FitzHugh–Nagumo model when using the data from the bottom left graph and a weakly informative (i.e., wide) prior term.

We target inverse problems to estimate parameters in an ordinary differential equation (ODE) that models the dynamically spiking membrane potential of a (biological) neuron. Spiking neurons in the brain and spinal cord are typically modeled by systems of nonlinear ODEs. These ODE models govern electrical voltage spikes that are generated in response to current stimuli. The output of one such ODE has the form of a spiking voltage time series, which can be obtained in laboratory experiments and takes the role of observational data in our inverse problem. These systems of ODEs contain uncertain parameters that control the opening and closing of ion channels of a cell membrane and consequently change voltage spike behavior.

Computational challenges arise from the highly nonlinear and nonconvex loss, or objective, function of the inverse problem (Buhry et al. (2008); see also the manifold in Figure 1, right), with sharp gradients, strong nonlinear dependencies between parameters, and potentially multiple local minima. The loss cannot be sufficiently regularized by a convex additive term stemming from a regularization or prior term, because of lack of knowledge about the range of parameter values (Gutenkunst et al., 2007; Prinz et al., 2004) and because this would largely eliminate the information from data and model. We therefore explore new techniques to solve such challenging inverse problems by fitting NN-based reconstruction maps that approximate solution operators for our inverse problem by means of mapping observational data to model parameters.

Contributions We consider neural networks with dense and convolutional layers and explore different NN architectures with a range of layers, dense units, and CNN filters. We quantify the capability of the networks to approximate reconstruction maps of inverse problems with several statistical metrics. Through these metrics, we aim to provide understanding about the effects of network choice on inference performance. The numerical results show that NNs can estimate parameters of spiking neuron models with high accuracy.

We investigate the predictive skills of NNs while changing the sizes of training sets. Additionally, we conduct experiments on partially observed data to numerically demonstrate that CNNs, but not dense NNs, are capable of detecting underlying properties or dynamics of a time series.

Moreover, since observational data are in practice most likely corrupted by noise, we present a noise model and analyze the influence of noise in training and/or testing data based on numerical experiments. To go beyond estimating ODE parameters, we extend our framework for NN-based reconstruction maps to simultaneously estimate parameters of the autocorrelated noise model from noisy observational data.

Related work The literature on parameter estimation for models of neural dynamics spans across various scientific communities, approaches, and neuron models. Here, we highlight only closely related publications and refer to the literature within these publications for further information. The neuron model by Hodgkin and Huxley (1952) is often used in the literature. It consists of a nonlinear system of four ODEs. The FitzHugh–Nagumo equations (introduced in Section 2.1) simplify the Hodgkin–Huxley model to a nonlinear system of two ODEs.

Common approaches for parameter estimation in FitzHugh–Nagumo and Hodgkin–Huxley models make use of heuristics and trial and error and may consist of intricate sequences of regression steps, as summarized by Buhry et al. (2008) and Van Geit et al. (2008). The techniques employed are, for instance, simulated annealing, differential evolution, genetic algorithms, and brute-force grid search (Alonso and Marder, 2019). These approaches have the disadvantage of being computationally expensive or are slow to converge. Using gradient descent for finding the minimizer of a Hodgkin–Huxley-based inverse problem (Doi et al., 2002), on the other hand, suffers from the strong nonlinearities in the objective and requires good initial guesses. Alternative approaches estimate parameters with maximum likelihood methods and build approximations of the likelihood term (Doruk and Abosharb, 2019). Recently, Kalman filters have been utilized, for instance, by Deng et al. (2009); ensemble Kalman filters are combined with reduced order models (Pagani et al., 2017), and augmented ensemble Kalman filters (Arnold and Lloyd, 2018) are employed for periodically time-varying parameters. In a data assimilation framework, Hamilton et al. (2018) propose an ensemble Kalman filter for time series with large amounts of noise. Furthermore, statistical inference in a Bayesian framework is attempted with approximate Bayesian computation (ABC) (Daly et al., 2015) and Monte Carlo sampling (Daly et al., 2018). Alternatively to time series data, Jolivet et al. (2006) and Naud et al. (2014) develop inference methods for spike time data.

While machine learning techniques generally are used for prediction and classification tasks in a wide range of applications, they are employed more sparsely to estimate parameters of mathematical models. For instance, Morshed and Kaluarachchi (1998) (for groundwater modeling) and Dua (2011) (for kinetik models) utilize NNs within their parameter estimation frameworks for systems described by partial or ordinary differential equations, they however do not consider approximating reconstruction maps. Parikh et al. (2020) employ generative adversarial networks to retrieve model parameters in stochastic inverse problems. Parameter estimation approaches for neural dynamics based on NNs are proposed (Gonçalves et al., 2020), where an NN is trained to perform posterior density estimates using mixtures of Gaussian or normalizing flows. While Gonçalves et al. (2020) show promising approximations of posterior features, their approach comes at the cost of large training sets (of order 10^5), which is about two orders of magnitude larger than in our work. Each training sample requires the solution of an ODE model and, if performed frequently, constitutes the major computational cost. For such large numbers of training samples, Monte Carlo methods (Ballnus et al., 2017) become preferable alternatives, especially if they could be augmented with reduced order models (Qian et al., 2020; O'Leary-Roseberry et al., 2020). Moreover, Gonçalves et al. (2020) do not analyze the effects of noise in training and/or testing data, and training set sizes are not varied. Their network architectures are prescribed, which reflects an inductive bias (Cranmer et al., 2020) by knowing which network performs well for a given problem.

In statistical models, machine learning is used for direct estimation of model parameters (Chon and Cohen, 1997), where NNs are used for estimating autoregressive moving-average processes parameters. Radev et al. (2020) propose a suite of two NNs to estimate summary statistics of the data, infer parameters of a model, and return samples from the posterior distributions. Other works focus on supplementing or complementing statistical inferential techniques with machine learning. In particular, problems with intractable likelihood are often solved with the ABC method involving the sampling of synthetic data and summary statistics of the observations. Creel (2017) and Jiang et al. (2017) uses an NN to predict the parameters from artificially generated data, which are then used as an estimate of the posterior mean of an ABC procedure.

The remainder of the paper is organized as follows. Section 2 introduces the forward problem governed by the FitzHugh–Nagumo model and the corresponding inverse problem. It then presents NN architectures utilized for solving the inverse problem, the generation of training and testing samples, and metrics to evaluate NN-based model parameter estimates. Section 3 discusses numerical results and quantifies prediction errors obtained from trained NNs. We consider a range of network architectures, different sizes of training data, and the effects of noise. Section 4 presents conclusions and open questions. Background information is collected in the Appendix.

2. Neural network-based reconstruction maps for inverse problems

This section introduces the forward problem that is governed by the FitzHugh–Nagumo ODE and the inverse problem to estimate model parameters of the ODE from time series data. Furthermore, we describe the NN architectures that will be used to perform the inference. We conclude with a set of metrics used to evaluate the errors of model parameter predictions coming from NNs.

2.1. Forward and inverse modeling of spiking neurons

The FitzHugh–Nagumo (FitzHugh, 1961; Nagumo et al., 1962) equations describe spiking neurons via a system of two ODEs,

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \gamma\left(u - \frac{u^3}{3} + v + \zeta\right), \tag{1a}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{1}{\gamma}\left(u - \theta_0 + \theta_1 v\right), \tag{1b}$$

where the unknowns of the ODE are the membrane potential $u = u(t)$ and the recovery variable $v = v(t)$. ζ denotes the total membrane current and is a stimulus applied to the neuron, which we assume to be constant in time. γ determines the strength of damping and is assumed to be known and constant. θ_0 and θ_1 are the model parameters that we consider for inference, because they govern two important characteristics of the oscillating solution of the ODE (1), namely spike rate and spike duration (see Appendix A for more details). ODE (1) is augmented with initial conditions $u(0) = u_0$, $v(0) = v_0$. An example solution of (1) is shown in the top left panel of Figure 1, using inference parameters $\theta_0 = 0.7$, $\theta_1 = 0.8$, constants $\gamma = 3.0$, $\zeta = -0.4$, and initial condition $u_0 = v_0 = 0$. While the FitzHugh–Nagumo model formulation is relatively simple compared with larger systems of ODEs, like the Hodgkin–Huxley equations, it exhibits most of the computational challenges when considered as the forward problem in an inference setting, such as described next.

We target the inverse problem, where we are given observational data

$$d(t) \coloneqq u_{\boldsymbol{\theta}^*}(t) + \eta(t), \tag{2}$$

composed of $u_{\boldsymbol{\theta}^*}(t)$, which is the first component (1a) of the solution of ODE (1) with unknown true parameters $\boldsymbol{\theta}^* \coloneqq (\theta_0^*, \theta_1^*)$, and added correlated noise $\eta(t)$ (see Section 3.4 for more information about the noise model). Our goal is to find parameters $\boldsymbol{\theta} \coloneqq (\theta_0, \theta_1)$ that are consistent with data $d(t)$ and model output $u_{\boldsymbol{\theta}}(t)$ for all t. The mathematical formulation for observational data as in (2) is commonly used (Cressie and Wikle, 2015) and does not preclude uniqueness of inferred parameters despite realizations of noise being nondeterministic. Note that the second variable v of the ODE (1) is excluded from the data because typically only the membrane potential can be observed in experiments. In a statistical/Bayesian framework (Kaipio and Somersalo, 2005), the inverse problem translates to finding the posterior probability density $\pi(\boldsymbol{\theta} \,|\, \boldsymbol{d})$ of the parameters $\boldsymbol{\theta}$ given data $\boldsymbol{d}$, where $\boldsymbol{d}$ is a discretization of $d(t)$. The posterior is, via Bayes' rule, composed of a likelihood term $\pi(\boldsymbol{d} \,|\, \boldsymbol{\theta})$ and a prior term $\pi(\boldsymbol{\theta})$,

$$\pi(\boldsymbol{\theta} \,|\, \boldsymbol{d}) \propto \pi(\boldsymbol{d} \,|\, \boldsymbol{\theta})\, \pi(\boldsymbol{\theta}). \tag{3}$$

In this work, we assume little knowledge about the range of parameter values $\boldsymbol{\theta}$, hence we target prior distributions that are merely weakly informative (i.e., relatively large variance of $\pi(\boldsymbol{\theta})$). In order to construct a weak prior, we consider that parameters of the FitzHugh–Nagumo model are in practice chosen to be inside the interval $\theta_0, \theta_1 \in [0, 1]$. Note that the parameter bounds are not required for solvability of the FitzHugh–Nagumo equations (1). Due to these bounds, our prior specifies that each parameter is normal and i.i.d. (independent identically distributed),

$$\theta_0 \sim \mathcal{N}\left(\bar{\theta}_{0,\text{prior}}, \sigma^2_{0,\text{prior}}\right) = \mathcal{N}\left(0.4, 0.3^2\right), \quad \theta_1 \sim \mathcal{N}\left(\bar{\theta}_{1,\text{prior}}, \sigma^2_{1,\text{prior}}\right) = \mathcal{N}\left(0.4, 0.4^2\right), \tag{4}$$

where $[0,1] \subset [\bar{\theta}_{i,\text{prior}} - 2\sigma_{i,\text{prior}}, \bar{\theta}_{i,\text{prior}} + 2\sigma_{i,\text{prior}}]$, $i = 0, 1$. Additionally, we limit the range of the parameters to be inside the intervals

$$\theta_0 \in [-0.2, 1.0] \quad \text{and} \quad \theta_1 \in [-0.4, 1.2] \tag{5}$$

by means of rejecting prior samples outside of these bounds. The prior bounds (5) restrict samples of $\boldsymbol{\theta}$ to ranges that contain the unit interval, and, moreover, they limit combinations of parameters where the FitzHugh–Nagumo model generates zero spikes or just one spike (see Appendix A).

For the NN-based solution techniques proposed in the following Section 2.2, we target point estimates of $\boldsymbol{\theta}$ directly. We observe an analogy in the context of Bayesian inference, since one type of point estimate of the posterior density is the maximum a posteriori (MAP) point (Calvetti and Somersalo, 2007; Stuart, 2010). The MAP point is obtained by minimizing the negative log of (3),

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\min_{\boldsymbol{\theta}} \Big[-\log\big(\pi(\boldsymbol{d} \,|\, \boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\big) \Big] = \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \left\| \frac{d(t) - u_{\boldsymbol{\theta}}(t)}{\sigma_{\text{noise}}} \right\|^2_{L_2} + \frac{1}{2} \left| \boldsymbol{\theta} - \bar{\boldsymbol{\theta}}_{\text{prior}} \right|^2_{\Sigma^{-2}_{\text{prior}}}, \tag{6}$$

where σ_{noise} denotes the standard deviation of the data noise, which is related to η in (2); $\bar{\boldsymbol{\theta}}_{\text{prior}} \coloneqq (\bar{\theta}_{0,\text{prior}}, \bar{\theta}_{1,\text{prior}})$ is the prior mean, and $\Sigma_{\text{prior}} \coloneqq \operatorname{diag}(\sigma_{0,\text{prior}}, \sigma_{1,\text{prior}})$ is the prior standard deviation. The resulting highly nonlinear loss function pertaining to minimization (6) is shown in Figure 1 (right), presenting significant computational challenges, in particular for gradient-based solvers. Note that our proposed NN-based reconstruction maps are not guaranteed to compute the MAP estimate or other known estimates, such as, maximum likelihood and conditional mean. The discussion of the MAP estimate serves as an illustration of computational challenges and as an analogy to interpret the results provided by NNs in Section 2.2.

2.2. Deep neural networks for inverse problems

We construct NNs to computationally learn reconstruction maps of parameter estimation problems. We set up the training data such that the inputs of the NN are time series (2) coming from the solution of the ODE (1) and the outputs are the corresponding parameters $\boldsymbol{\theta}$ that generated the time series. Since we train the NN to fit a mapping of time series to parameters in the reverse order of the forward problem, the NN is learning to represent a pseudoinverse of the forward operator (Adler and Öktem, 2017; Fan et al., 2019; Khoo and Ying, 2019), hence approximately solving an inverse problem. We define our NN-based reconstruction map as

$$\hat{\boldsymbol{\theta}} := y_L, \quad y_\ell = \mathcal{F}_\ell(y_{\ell-1}) \text{ for } 1 \le \ell \le L, \quad y_0 := \boldsymbol{d}, \tag{7}$$

where the NN input $\boldsymbol{d}$ is a discrete version of the time series (2), $\mathcal{F}_\ell$ denotes layer ℓ of the NN, and $\hat{\boldsymbol{\theta}}$ is the network's output, which are predicted model parameters.

Training and testing data for neural networks We generate training and testing data for NNs by sampling parameters $\boldsymbol{\theta}$ from their prior distributions (4) and discarding samples outside of prior bounds (5). By choosing samples from the prior to generate training data, we want the NN reconstruction (7) to learn the most important features of our inverse problem according to prior knowledge. Furthermore, using the prior for generating training data creates a framework that is analogous to the original Bayesian inverse problem (3). Having obtained samples of $\boldsymbol{\theta}$, we solve the ODE (1) and store the membrane potential $u_{\boldsymbol{\theta}}(t_i)$ at prescribed time steps, t_i, $i = 1, \dots, N_t$, with uniform step size Δ_t. The next two paragraphs describe the different architectures for the layers denoted by $\mathcal{F}_\ell$ in Equation (7).

Dense neural network (dense NN) Our first NN architectures consist of a sequence of dense, or fully connected, layers (Goodfellow et al., 2016). Each dense layer ℓ consists of $n_u = n_u(\ell)$ units, or nodes, and each unit of one layer is connected to all other units of a neighboring layer. One dense layer is composed of an affine mapping and a nonlinear function, $\mathcal{F}_\ell(y_{\ell-1}) = \phi(W_\ell\, y_{\ell-1} + b_\ell)$, where the matrix $W_\ell \in \mathrm{R}^{n_u(\ell)\times n_u(\ell-1)}$ are the weights and $b_\ell \in \mathrm{R}^{n_u(\ell)}$ is the bias of layer ℓ. The nonlinear activation function ϕ is applied element-wise. A typical choice for ϕ is the rectified linear unit (ReLU) function (Schmidhuber, 2015). We, however, utilize a smoother activation similar to ReLU, called Swish (Elfwing et al., 2018). NNs with Swish activation have been shown to suffer less from vanishing gradient problems compared with ReLU (Hayou et al., 2018), and this behavior has been observed by the authors when training deeper networks. For our numerical experiments in Section 3, we use dense NNs with 2–16 layers and 4–128 units n_u in all layers to cover a wide range of network sizes, which we will demonstrate to be sufficient for accurate approximation of reconstruction maps. The dense NNs amount to a total of between 4,034 and 442,114 trainable NN parameters in the form of weight matrices W_ℓ and biases b_ℓ, which are optimized with stochastic gradient descent by using the Adam algorithm (Kingma and Ba, 2015) and the mean squared error (MSE) loss function (see Section 3.1 for more details).

Convolutional neural network (CNN) CNNs have been successfully used in two-dimensional image-processing tasks (LeCun et al., 1999). We take advantage of the local dependence structure inherent in CNNs to fit reconstruction maps that have one-dimensional time series with uniform time steps as input. We therefore exploit locality in our time series with convolutional layers. A convolutional layer is composed of n_f filters, and each filter is associated with one kernel that is applied to small sections of the time series. The n_f kernels of one convolutional layer are connected to all neighboring layers, similarly to the units of dense layers described above. The weights of these connections constitute the NN parameters to be optimized. Because the weights are shared across time, significantly fewer weights have to be optimized.

Our convolutional architectures borrow from CNNs for image classification (Krizhevsky et al., 2012), where we interleave convolutional layers with pooling layers (also called downsampling), which aggregate a small block from a convolutional step into a single value. Each convolutional layer, $y_\ell = \mathcal{F}_\ell(y_{\ell-1})$, reduces the size of output vectors y_ℓ compared with input $y_{\ell-1}$, here by a factor of 1/2; the same reduction holds for pooling layers. The reduction in size is complemented with an increase of n_f filter counts, here by a factor of 2, from one convolutional layer to the next. After a few combinations of convolution and pooling, the output is passed to a sequence of dense layers, which concludes the CNN. Our CNNs are equipped with 2–4 pairs of convolution and pooling layers with varying numbers of filters (see Section 3.2 for more details), and we keep the dense layers at the end of the network fixed to two layers with $n_u = 32$ units. These CNNs achieve NN parameter counts between 5,278 and 261,442. We utilize the same Swish activation function and optimization setup as for dense NNs described above.

2.3. Evaluation metrics for inference results

To assess the quality of the prediction from NNs, we carry out both qualitative and quantitative evaluations in Section 3. Qualitative assessments are done through scatterplots and time series plots, and the quantification of prediction capabilities is evaluated with several metrics.

In particular, we decompose the commonly used mean squared error (MSE) into its squared bias and centered MSE (C-MSE) components to assess the respective contributions of the mean and of the fluctuations (variability) of the prediction mismatch (Taylor, 2001):

$$\mathrm{MSE}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) := \frac{1}{M}\sum_{j=1}^{M}\left(\boldsymbol{\theta}^{(j)} - \hat{\boldsymbol{\theta}}^{(j)}\right)^2 = \overbrace{\left(\bar{\boldsymbol{\theta}} - \bar{\hat{\boldsymbol{\theta}}}\right)^2}^{=\text{Squared bias}} + \overbrace{\frac{1}{M}\sum_{j=1}^{M}\left[\left(\boldsymbol{\theta}^{(j)} - \bar{\boldsymbol{\theta}}\right) - \left(\hat{\boldsymbol{\theta}}^{(j)} - \bar{\hat{\boldsymbol{\theta}}}\right)\right]^2}^{=\text{C-MSE}},$$

where $\boldsymbol{\theta}$ is the true model parameter, $\hat{\boldsymbol{\theta}}$ is the predicted parameter, $\bar{\boldsymbol{\theta}}$ (resp. $\bar{\hat{\boldsymbol{\theta}}}$) is the mean of true parameters (resp. estimated parameters), and M is the training set size. We use superscript notation (e.g., $\boldsymbol{\theta}^{(j)}$) to identify different parameters in the training set. A smaller MSE implies lower errors and hence better predictions.

Since MSE has the disadvantage of being sensitive to large errors, we report the median of the absolute percentage error (Median-APE), $\left|\boldsymbol{\theta}^{(j)} - \hat{\boldsymbol{\theta}}^{(j)}\right| / \left|\boldsymbol{\theta}^{(j)}\right|$. This also provides a scale-invariant metric that is less sensitive to outliers than a mean statistic is. The coefficient of determination R^2 is used to assess the percentage of variability explained by the prediction $\mathrm{R}^2 := 1 - \sum_{j=1}^{M}\left(\boldsymbol{\theta}^{(j)} - \hat{\boldsymbol{\theta}}^{(j)}\right)^2 \Big/ \sum_{j=1}^{M}\left(\boldsymbol{\theta}^{(j)} - \bar{\boldsymbol{\theta}}\right)^2$. The R^2 takes values between $-\infty$ and 1; the closer R^2 is to the ideal value 1, the more variability is explained by the prediction.

3. Numerical experiments

In this section we numerically analyze the accuracy of model parameter estimates from dense NNs and CNNs defined in Section 2.2. We investigate the effects of different network architectures on the prediction errors of FitzHugh–Nagumo model parameters from noise-free time series data, and we analyze the sensitivity of predictions in the presence of noise in training and/or testing data. For these numerical studies, we utilize the metrics described in Section 2.3.

Table 1: Settings for training/testing data and optimization algorithms.

Training & optimization setting	Value
Number of training samples (N)	1000
Number of testing samples (M)	2000
Number of validating samples	2000
Size of time series (N_t)	1000
Loss function	MSE
Optimizer	Adam
Learning rate (or step length)	0.002
Batch size	32
Number of epochs with noise-free data	200
Number of epochs with noisy data	50

Table 2: Settings for dense and convolutional neural network architectures.

Neural network setting	Value
Activation function (dense NN & CNN)	Swish
CNN kernel size	3
CNN kernel stride	2
CNN pooling type	average
CNN pooling size	2
CNN pooling stride	2
CNN padding	none
CNN flattening into dense layers	2 layers, $n_u = 32$

3.1. Experimental setup

The implementation of our algorithms is carried out in Python. We use the explicit Runge–Kutta method of order 3(2) (RK23) for time integration of the FitzHugh–Nagumo ODE, and we utilize TensorFlow/Keras (version 2.3.1) for implementing the neural networks. To generate data sets for training, validating, and testing the networks, we simulate the ODE (1) for 200 milliseconds and store the membrane potential at equidistant steps every $\Delta_t = 0.2$, which results in a time series of size $N_t = 1000$ for each model parameter $\boldsymbol{\theta} \in \mathrm{R}^2$. Additionally, we use the Fourier transform of the time series for training and prediction in Sections 3.3 and 3.5. We randomly generate sets of training, validating, and testing data based on the prior of $\boldsymbol{\theta}$ as described in Section 2. Training is performed with $N = 1000$ samples unless otherwise specified. 2000 samples are used for validation and testing is carried out with a sample size of $M = 2000$. Table 1 provides a summary of the experimental settings regarding the optimization, and Table 2 shows settings for network architectures.

3.2. Exploration of neural network architectures

In the following, we investigate the influence of different NN architectures from Section 2.2 on predictions performed on the validation data set. We explore effects on predictive skills while varying the number of dense layers and number of units per layer, in the case of dense NNs, and the number of convolutional layers and filters per layer, in the case of CNNs. The training of the networks as well as predictions are performed on noise-free time series data, which represents an ideal scenario for parameter estimation. Noise-free data have the advantage that optimization algorithms, using stochastic gradient descent, can run for sufficiently high iteration counts (or epochs), without leading to overfitting due to noise in the data. This allows us to train the networks relatively well and to investigate differences in predictions that are largely due to network architectures. Prediction results with noisy observational data are crucial in practice and are presented in Section 3.4.

The results in this section are augmented with cross-validation experiments and a study on the sensitivity on the initialization of NN parameters for selected NN architectures in Appendix B.

Model parameter predictions with dense NNs The prediction capability of dense NNs are evaluated with the squared bias and C-MSE metrics in Table 3 for the number of dense layers varying from 2 to 16 and the number of units per layer n_u between 4 and 128. Corresponding evalua-

Table 3: Squared bias (C-MSE) of model parameter predictions (noise-free), using **dense NNs** with variable numbers of layers and units per layer n_u.

n_u	**2 layers**	**4 layers**	**8 layers**	**12 layers**	**16 layers**
4	1.2×10^{-4} (0.0154)	1.0×10^{-5} (0.0085)	3.6×10^{-4} (0.0209)	5.0×10^{-4} (0.0304)	1.0×10^{-4} (0.0269)
8	1.2×10^{-4} (0.0031)	1.0×10^{-4} (0.0059)	2.8×10^{-4} (0.0024)	3.2×10^{-5} (0.0036)	1.6×10^{-4} (0.0225)
16	5.2×10^{-5} (0.0023)	6.2×10^{-5} (0.0023)	8.6×10^{-5} (0.0017)	1.6×10^{-5} (0.0030)	1.3×10^{-4} (0.0033)
32	8.4×10^{-6} (0.0020)	3.7×10^{-5} (0.0020)	1.1×10^{-5} (0.0017)	2.5×10^{-5} (0.0019)	2.6×10^{-5} (0.0025)
64	1.1×10^{-4} (0.0026)	6.4×10^{-5} (0.0022)	9.8×10^{-5} (0.0018)	2.0×10^{-4} (0.0033)	8.6×10^{-6} (0.0026)
128	5.3×10^{-4} (0.0090)	9.7×10^{-6} (0.0015)	4.9×10^{-5} (0.0022)	2.3×10^{-5} (0.0026)	1.1×10^{-4} (0.0023)

Table 4: Median-APE (R^2) of model parameter predictions (noise-free), using **dense NNs**.

n_u	**2 layers**	**4 layers**	**8 layers**	**12 layers**	**16 layers**
4	0.176 (0.835)	0.065 (0.916)	0.103 (0.793)	0.239 (0.693)	0.203 (0.732)
8	0.059 (0.968)	0.051 (0.944)	0.049 (0.973)	0.042 (0.965)	0.182 (0.787)
16	0.040 (0.976)	0.044 (0.975)	0.032 (0.982)	0.029 (0.971)	0.046 (0.967)
32	0.029 (0.979)	0.025 (0.979)	0.026 (0.983)	0.026 (0.980)	0.059 (0.974)
64	0.043 (0.972)	0.029 (0.977)	0.038 (0.980)	0.063 (0.967)	0.045 (0.974)
128	0.082 (0.905)	0.019 (0.985)	0.035 (0.977)	0.051 (0.973)	0.052 (0.976)

tions with the Median-APE and R^2 metrics are shown in Table 4. The tables highlight in red those configurations leading to relatively poor predictions and in green those leading to relatively good predictions. Overall, we observe a low error in average squared bias (below 10^{-3}) and mild variations of the errors, expressed in C-MSE values below 10^{-2}, showing that a large contribution to the total MSE is due to the mismatch in variability between true and predicted parameters. Additionally, the scale invariant metric Median-APE assumes values mostly below 10^{-1} and the R^2 metric mostly above 0.95. From these tables, we can deduce that networks with 2–12 layers and 16–64 units per layer lead to the lowest prediction errors (green cells in Tables 3, 4). Networks with small unit counts, such as 4, however, deteriorate in their prediction skills, because they do not offer sufficient degrees of freedom to represent a reconstruction map. Furthermore, deeper networks with large numbers of layers, such as 16, do not offer any improvements of prediction errors compared with shallower architectures. Due to these results, we select the dense NN architecture with 4 layers and $n_u = 32$ units per layer for cross-validation and inspection of sensitivity to initialization of NN parameters (see Appendix B); and we carry out subsequent experiments with this choice of dense NN. Next, we improve upon the already good results of dense NNs using convolutional layers.

Model parameter predictions with CNNs Analogous to the results with dense NNs, we now consider CNNs while changing the number of convolutional layers and the number of filters n_f per layer. Two dense layers with 32 units follow the convolutional layers and remain fixed during these experiments. Table 5 shows evaluations with squared bias and C-MSE metrics, and Table 6 considers the Median-APE and R^2 metrics. For brevity, we express the convolutional layers as a list, for instance $n_f \times [1, 2, 4]$, which denotes $(n_f \times 1)$ filters in the first layer, $(n_f \times 2)$ filters in the second layer, and $(n_f \times 4)$ filters in the third layer. Tables 5 and 6 show that CNNs can further reduce the prediction errors for our inverse problem compared with dense NNs. This improvement is demonstrated by C-MSE values below 10^{-3} (one order of magnitude lower than with dense NNs) and R^2 values mostly above 0.99 (previously 0.95–0.98 with dense NNs), which is very close to the

Table 5: Squared bias (C-MSE) of model parameter predictions (noise-free), using **CNNs** with variable numbers of convolutional layers and filters per layer n_f.

n_f	$n_f \times [1, 2]$	$n_f \times [1, 2, 4]$	$n_f \times [1, 2, 4, 8]$
2	5.2×10^{-5} (0.000 89)	1.3×10^{-5} (0.000 61)	6.6×10^{-7} (0.000 56)
4	4.5×10^{-5} (0.000 69)	4.2×10^{-6} (0.000 48)	2.7×10^{-5} (0.000 45)
8	3.0×10^{-7} (0.000 71)	1.8×10^{-5} (0.000 48)	2.0×10^{-5} (0.000 44)
16	2.2×10^{-5} (0.000 66)	4.5×10^{-5} (0.000 54)	3.0×10^{-5} (0.000 65)
32	1.7×10^{-5} (0.001 08)	2.5×10^{-4} (0.000 64)	1.4×10^{-4} (0.000 51)

Table 6: Median-APE (R^2) of model parameter predictions (noise-free), using **CNNs**.

n_f	$n_f \times [1, 2]$	$n_f \times [1, 2, 4]$	$n_f \times [1, 2, 4, 8]$
2	0.022 (0.990)	0.017 (0.993)	0.017 (0.994)
4	0.023 (0.992)	0.016 (0.995)	0.013 (0.995)
8	0.017 (0.992)	0.018 (0.994)	0.016 (0.995)
16	0.018 (0.993)	0.022 (0.993)	0.026 (0.993)
32	0.021 (0.988)	0.032 (0.991)	0.033 (0.993)

ideal R^2 of 1.0. These improvements relative to dense NN can be attributed to the CNNs exploiting local dependence information of neighboring points in the time series.

We additionally observe that CNNs with only two convolutional layers yield higher prediction errors than with three and four layers. Thus the short networks do not offer sufficient degrees of freedom to adapt to the reconstruction map of the inverse problem. Moreover, increasing the number of filters to $n_f = 16$ and beyond shows no benefit for predictive skills. The best results are achieved with 3–4 convolutional layers and $n_f = 4, \ldots, 8$, which we highlight with green in the tables. Because of the results, we select the CNN consisting of 3 convolutional layers with 8, 16, and 32 filters, in short denoted by $n_f \times [1, 2, 4]$, $n_f = 8$, for cross-validation and inspection of sensitivity to initialization of NN parameters (see Appendix B); and we carry out subsequent numerical experiments with this choice of CNN.

3.3. Learning capabilities of neural networks for partially observed time series

This section demonstrates numerically how the networks are learning as we try to find answers to the higher-level question: Is a NN merely "remembering" a time series or is it learning underlying properties or dynamics? To this end, we design an experiment where the time series of the training data from Section 3.1 is split into two halves of length $N_t/2 = 500$, hence doubling the number of training samples from $N = 1000$ to 2000. After training, the predictions are performed on time series of the testing data that are also of length $N_t/2 = 500$. However, we extract the following five arbitrary overlapping intervals from the original testing data of length $N_t = 1000$ (see Section 3.1): $[30, 530)$, $[146, 646)$, $[174, 674)$, $[362, 862)$, $[370, 870)$; thus increasing the amount of testing samples from $M = 2000$ to 10 000. In addition to using time series data, we use a second input data type by replacing each time series by its Fourier transform; and a third input data type by combining time series and Fourier transform. The results in Tables 7 and 8 clearly show the advantage of CNNs over dense NNs. While the CNN's accuracy in this setting is similar to the one presented in Section 3.2, the dense NN performs poorly ($R^2 < 0.5$ with time series data). The dense network's

Table 7: Predictions with partially observed time series (noise-free), using a **dense NN** with 4 layers, $n_u = 32$; performance is poor compared with CNNs.

Data type	**Sq. bias**	**C-MSE**	**Med.-APE**	**R^2**
Time	3.51×10^{-3}	0.0534	0.2632	0.475
Fourier	2.48×10^{-4}	0.0312	0.1478	0.620
Time & Fourier	4.27×10^{-3}	0.0468	0.2503	0.528

Table 8: Predictions with partially observed time series (noise-free); using a **CNN** with $n_f \times [1, 2, 4]$, $n_f = 8$; performance similar as with full time series.

Data type	**Sq. bias**	**C-MSE**	**Med.-APE**	**R^2**
Time	8.38×10^{-5}	0.003 34	0.0235	0.970
Fourier	1.70×10^{-4}	0.024 56	0.1235	0.685
Time & Fourier	8.59×10^{-6}	0.002 22	0.0289	0.980

prediction capability improves when Fourier data is used; however, the model parameter predictions from a Fourier spectrum are significantly worse compared with previous results using (full) time series data (Tables 3, 4). Overall, we conclude from these results that only CNNs demonstrate a capability to extract crucial properties of the time series that are important for inferring model parameters from arbitrary, partial observations. This is presumably achieved by the shift-invariant kernel of convolutional layers. Thus, the results suggest that CNNs not simply memorize patterns, but rather recognize properties or dynamics of the ODE.

3.4. Parameter estimation in the presence of noise in observational data

Noise model We consider an additive noise model that is first-order autoregressive in time and widely used for representing time series. The autoregressive process is parametrized by a correlation parameter ρ, which determines the dependence of the process on its previous value (Mills, 1991). Recall from (2) that the observational data are $d(t) = u_{\boldsymbol{\theta}^*}(t) + \eta(t)$, where $u_{\boldsymbol{\theta}^*}(t)$ comes from the solution of (1) and $\eta(t)$ is a correlated noise that evolves in time as

$$\eta(t_i) \coloneqq \rho\, \eta(t_{i-1}) + \epsilon(t_i), \quad i = 2, \dots, N_t, \quad \eta(t_1) \sim \mathcal{N}\Big(0, \tfrac{\sigma^2}{\Delta_t^2}\Big) \tag{8}$$

with $|\rho| < 1$. The term $\epsilon(t_i)$ is independent from $\eta(t_{i-1})$ and the process ϵ is a normally distributed white noise. To reflect the model resolution Δ_t in the level of variance in noise η, we prescribe $\text{var}(\eta) = \sigma^2/\Delta_t^2$ together with ρ. The variance of η is constant across time since the process is stationary due to $|\rho| < 1$, meaning that the stationary distribution of the process is $\eta(t) \sim \mathcal{N}\big(0, \frac{\sigma^2}{\Delta_t^2}\big)$ for all t. Consequently, we derive[1] $\epsilon(t_i) \sim \mathcal{N}\big(0, \frac{\sigma^2}{\Delta_t^2}(1 - \rho^2)\big)$.

In the following, when noisy data are used, the values of the parameter ρ and σ are varied randomly across different samples of (2). We generate 100 independent samples of

$$\rho \sim \mathcal{N}(0.8, 0.05^2) \quad \text{and} \quad \sigma \sim \mathcal{N}(0.07, 0.01^2) \tag{9}$$

in order to achieve a correlation ρ of 0.65–0.95 and a Δ_t-independent noise level of 4–10% via σ, respectively. For each value of the pair (ρ, σ), independent replicates of the process η following (8) are generated and added to training and/or testing data.

1. Note that it is more common to prescribe $\text{var}(\epsilon)$ together with ρ first and then derive $\text{var}(\eta)$; however, we proceed in the reverse order because we are foremost interested in defining the effective variance of η.

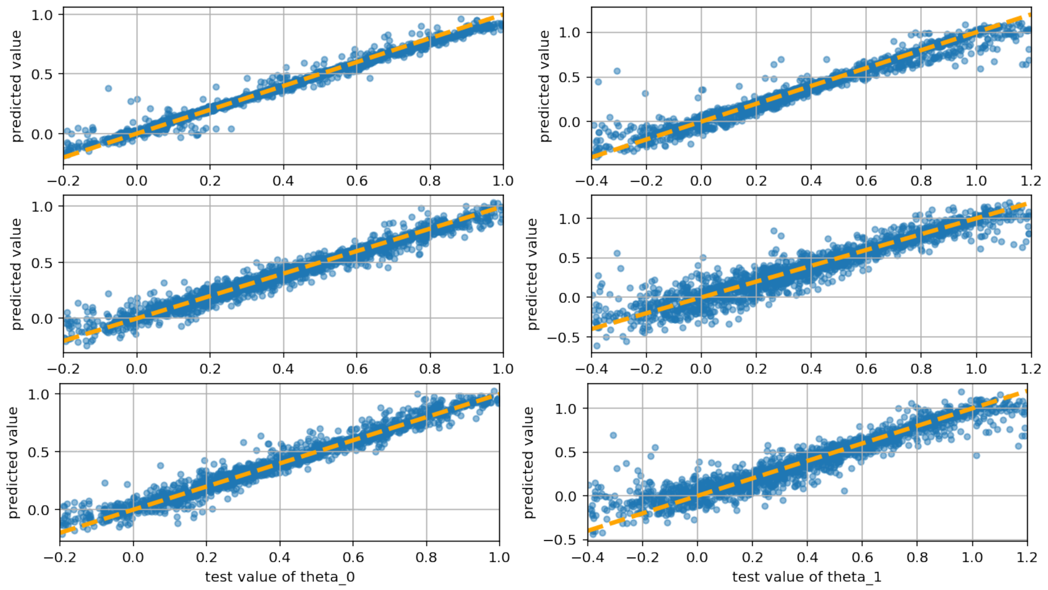

Figure 2: Predictions of parameters θ_0 (left column) and θ_1 (right column) from noise-free and noisy observational data, using a **dense NN** (4 layers, $n_u = 32$) and $N = 1000$ training samples. Top row: training and testing data noise-free; Middle row: training data noise-free and testing data with noise; Bottom row: training and testing data with noise. Dashed orange line depicts perfect predictions.

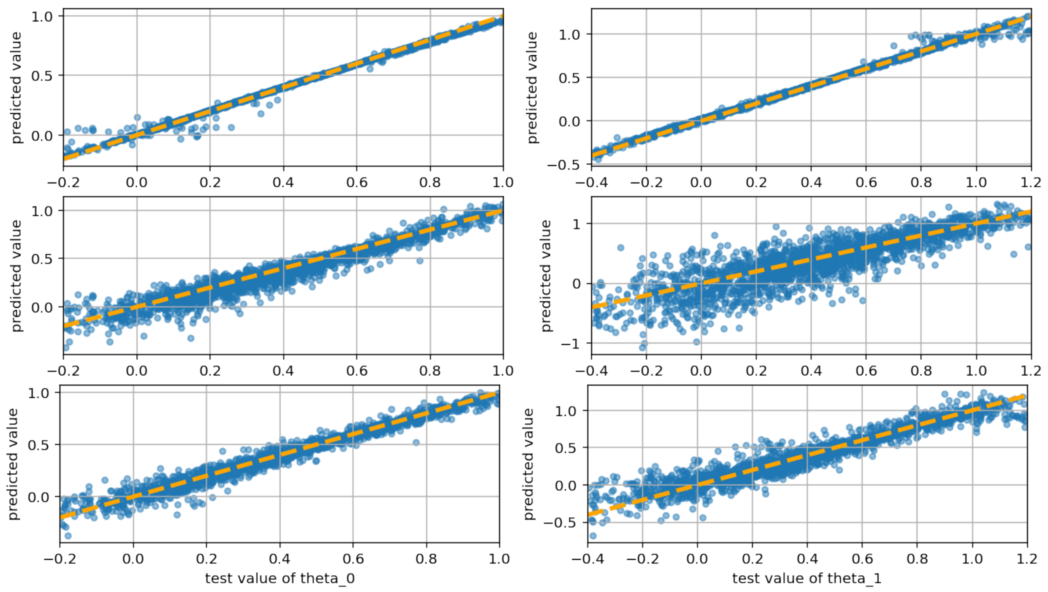

Figure 3: Predictions of parameters θ_0 (left column) and θ_1 (right column) from noise-free and noisy observational data, using a **CNN** ($n_f \times [1, 2, 4]$, $n_f = 8$) and $N = 1000$ training samples. Top row: training and testing data noise-free; Middle row: training data noise-free and testing data with noise; Bottom row: training and testing data with noise. Dashed orange line depicts perfect predictions.

Table 9: Median-APE (R^2) of model parameter predictions with noisy observational data, using a **dense NN** with 4 layers, $n_u = 32$.

N	**train noise-free test noise-free**	**train noise-free test with noise**	**train with noise test with noise**
500	0.043 (0.960)	0.098 (0.914)	0.105 (0.879)
1000	0.021 (0.978)	0.103 (0.918)	0.082 (0.921)
4000	0.014 (0.993)	0.089 (0.927)	0.061 (0.961)
8000	0.021 (0.992)	0.098 (0.921)	0.062 (0.968)

Table 10: Median-APE (R^2) of model parameter predictions with noisy observational data, using a **CNN** with $n_f \times [1, 2, 4]$, $n_f = 8$.

N	**train noise-free test noise-free**	**train noise-free test with noise**	**train with noise test with noise**
500	0.023 (0.990)	0.169 (0.788)	0.098 (0.921)
1000	0.014 (0.995)	0.174 (0.763)	0.096 (0.938)
4000	0.014 (0.997)	0.204 (0.710)	0.060 (0.970)
8000	0.014 (0.998)	0.251 (0.617)	0.053 (0.976)

Model parameter predictions with noise More relevant in practice are model parameter estimates in the presence of noise in observational data. Using one dense NN and one CNN, we demonstrate the effects of the additive noise stemming from the model described above on predictive skills. For these experiments, we use a dense NN with four layers and $n_u = 32$ units per layer, and the CNN has 3 convolutional layers with $n_f \times [1, 2, 4]$, $n_f = 8$ and 2 dense layers. Tables 9 and 10 show the prediction errors for the dense NN and CNN, respectively. Three columns in each table represent different noise configurations: both training and testing data do not contain noise (second column); only testing data contain noise (third column); and both training and testing data contain noise (fourth column). The tables additionally demonstrate the effects of using smaller and larger amounts of training data, denoted by N (first column), indicating in most cases improved predictive skills with increasing training sets. Throughout all experiments, the testing data of size $M = 2000$ are kept fixed to allow for fair comparisons.

For both networks, Tables 9 and 10 show the lowest prediction errors for noise-free training and testing data, whereas the highest prediction errors occur when the training data do not contain noise but the test data do have noise. For CNNs, predictions with noise in testing data improve significantly if the training data are also noisy (Table 10, last column). These results show the importance of training CNNs with noisy data in practice, even if noise-free simulated model outputs are available, because data from experiments or measurements are very likely polluted by some amount of noise. Comparing the two network architectures, dense NN and CNN, the CNN delivers lower prediction errors in most cases except when the training of the CNN is performed with noise-free data while testing data have noise. Only in this case, the dense NN demonstrates significantly better prediction skills (dense NN has $R^2 > 0.92$ vs. CNN has $R^2 < 0.90$) and therefore shows that dense layers are less sensitive to the presence or absence of noise in the training data.

We complete the discussion about prediction errors in the presence of observational noise by presenting scatterplots in Figures 2 and 3 describing the distribution of errors for varying values of model parameters θ_0 and θ_1. Each graph in the figures shows the true value of θ_0 or θ_1 from the test data set on the horizontal axis and the corresponding predicted value on the vertical axis. Deviations of the predictions (blue dots) from the ideal (orange line) show the prediction errors. The figures support the summarized statistics in Tables 9 and 10 discussed above and additionally show that the spread of prediction errors is largely uniform across values of θ_0 and θ_1. Slightly larger errors can appear at the lower and upper bounds of θ_0 and θ_1; these are more pronounced for the dense NN than for the CNN.

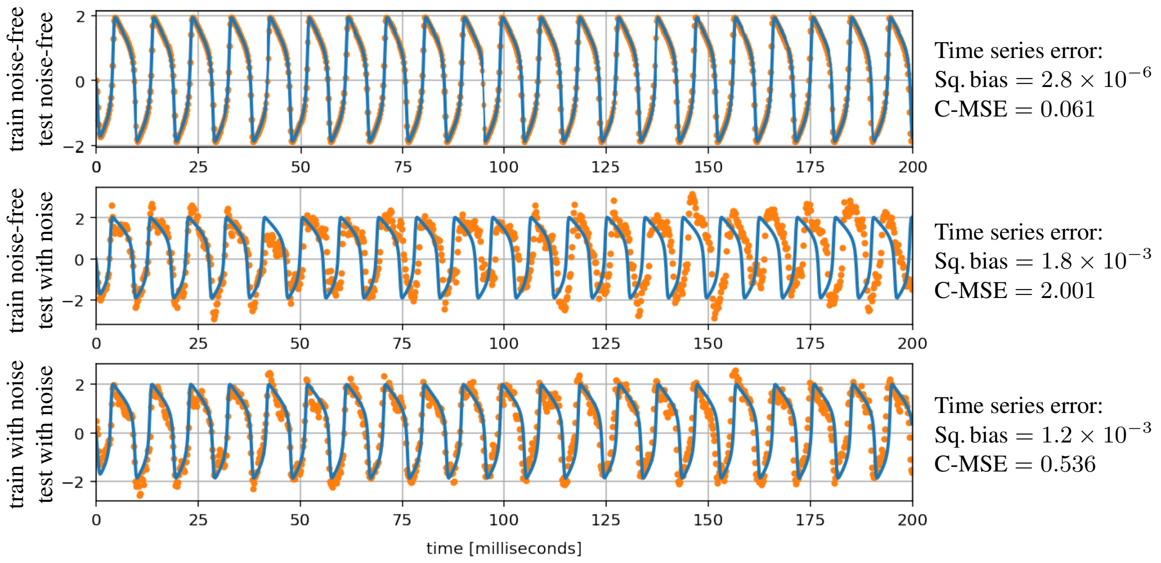

Figure 4: Simulations of FitzHugh–Nagumo model (blue lines) using parameters from CNN predictions; corresponding data that gave rise to prediction are shown as orange dots. Each graph shows the median percentile of MSE for the following cases. Top: training and testing data noise-free; Middle: training data noise-free and testing data with noise; Bottom: training and testing data with noise.

Simulations of FitzHugh–Nagumo model from predicted parameters We extend the evaluation of results on model parameter predictions and now evaluate simulations of the FitzHugh–Nagumo model (1) with predicted parameters. The results presented here show the errors of the neural network predictions propagated through the forward problem. We focus on the CNN with $n_f \times [1, 2, 4]$, $n_f = 8$, and we consider the same three different cases of presence and/or absence of noise as before. Figure 4 shows three FitzHugh–Nagumo simulations, in which the top graph represents the case where both training and testing data do not contain noise; in the middle graph only testing data contain noise; and the bottom graph shows results where both training and testing data contain noise. Each of the three graphs is selected from the testing data set as the median of the MSE between true and predicted parameters. More detailed results for FitzHugh–Nagumo simulations from predicted parameters are given in Appendix C, where we show results for the 10th, 25th, 75th, and 90th percentiles of MSE between simulated and testing time series.

We observe a nearly optimal overlapping of simulated output from predicted parameters and test data corresponding to "true" parameters for the noise-free case (Figure 4, top). When noise is added to training or testing data, the simulated time series' show shifted spikes, which become more pronounced as simulation time increases (100–200 milliseconds). The shifting of spikes over time represents discrepancies with respect to the frequency of a periodic time series. This effect is more pronounced when the training data is noise free (Figure 4, middle) and only mildly visible when the training data has noise (Figure 4, bottom). Overall, the results here and in Appendix C quantify the accuracy and reliability of the proposed estimation in order to generate realistic solutions of the FitzHugh–Nagumo model from estimated parameters with convolutional networks.

Table 11: Median-APE (R^2) of joint parameter predictions of the ODE and noise parameters, using a CNN with $n_f \times [1,2,4]$, $n_f = 8$. Providing Fourier spectrum to network becomes crucial when inferring noise parameters, yielding best results for time & Fourier data.

N	**Data type**	**FitzHugh–Nagumo parameter**		**Noise parameter**	
		θ_0	θ_1	σ	ρ
500	Time	0.126 (0.897)	0.230 (0.778)	0.110 (−0.76)	0.064 (−0.79)
	Fourier	0.258 (0.449)	0.349 (0.519)	0.071 (0.416)	0.032 (0.539)
	Time & Fourier	0.103 (0.921)	0.209 (0.793)	0.063 (0.549)	0.030 (0.606)
1000	Time	0.115 (0.914)	0.213 (0.812)	0.113 (−0.62)	0.063 (−0.80)
	Fourier	0.243 (0.460)	0.315 (0.577)	0.064 (0.524)	0.028 (0.603)
	Time & Fourier	0.103 (0.935)	0.192 (0.856)	0.058 (0.589)	0.028 (0.645)
4000	Time	0.089 (0.949)	0.168 (0.907)	0.068 (0.464)	0.050 (−0.15)
	Fourier	0.230 (0.517)	0.259 (0.748)	0.056 (0.611)	0.027 (0.653)
	Time & Fourier	0.076 (0.962)	0.136 (0.916)	0.053 (0.657)	0.026 (0.700)
8000	Time	0.070 (0.962)	0.138 (0.933)	0.058 (0.627)	0.030 (0.557)
	Fourier	0.162 (0.580)	0.215 (0.797)	0.051 (0.669)	0.023 (0.721)
	Time & Fourier	0.066 (0.968)	0.110 (0.942)	0.050 (0.684)	0.024 (0.722)

3.5. Joint estimation of FitzHugh–Nagumo parameters and noise parameters

In addition to inferring parameters of the FitzHugh–Nagumo ODE, we are also interested in inferring properties of the noise contained in a time series. Therefore, we target the simultaneous estimation of model and noise parameters with a single NN in this section. This means that we extend the output space of the NN-based reconstruction map from two model-parameters (θ_0, θ_1) to four parameters $(\theta_0, \theta_1, \sigma, \rho)$, where σ is the noise standard deviation and ρ is the autocorrelation parameter of the noise model (8). Such a joint estimation of parameters of (deterministic) physical models based on differential equations and, simultaneously, of parameters of statistical models is challenging because of the extremely different time scales in the physical vs. noise stochastic processes. Due to these difficulties, joint parameter estimation governed by physical and statistical models are rarely attempted in the literature.

We utilize the framework for computationally learning reconstruction maps described in Section 2. Our framework requires samples for training and testing from a prior distribution; therefore we augment the prior for ODE model parameters θ_0, θ_1 from Equation (4) with the prior information (9) for the noise parameters σ, ρ. In order to address the extreme range of time scales between ODE and noise models, we employ the Fourier transform of a noisy time series. This yields three different types of observational data that can be used to train a network and perform predictions: a time series, a Fourier spectrum, and a time series combined with its Fourier spectrum. We consider the three data types as well as different sizes of training data sets in our numerical experiments for joint ODE and noise parameter estimation.

In practice, it is necessary to preprocess the different types of data with scaling transformations in order to rescale time series and Fourier transforms as well as the ODE and noise parameters; this can be carried out with standard libraries (e.g., scikit-learn). The results are presented in Table 11 for a CNN consisting of 3 convolutional layers with $n_f \times [1,2,4]$, $n_f = 8$ followed by 2 dense layers. The crucial role of the Fourier spectrum in order to estimate noise parameters is clearly demonstrated by negative R^2 values where only time series data is provided to the CNN. If spectrum data is omitted, very large amounts of training data are necessary ($N = 8000$ in this case) in order to achieve a merely non-negative R^2 for σ, ρ. On the other hand, the Fourier spectrum by itself yields significantly larger errors for ODE parameters compared with time series data. As a consequence, the best prediction performance for all four parameters is achieved when the CNN is trained with both a time series and the corresponding Fourier spectrum. Overall, our framework for computationally learning NN-based reconstruction maps has demonstrated accurate predictions of model parameters, the reconstruction maps are able to handle noisy data very well, and previously challenging or even infeasible inference results become within reach as shown for the joint estimation of ODE and noise parameters.

4. Conclusion

In this paper, we build an estimation framework for an inverse problem governed by an ODE, the FitzHugh–Nagumo model. The estimation consists of recovering model parameters as a prediction output from neural networks that are trained on time series solutions of the model as input. Our study shows the efficacy of neural network-based parameter recovery when traditional optimization techniques would fail to minimize the misfit function of the problem. In particular, we propose and compare a range of different architectures of dense and convolutional NNs in order to computationally learn reconstruction maps of the inverse problem in different situations (ideal train and test data, noisy train and test data, and noisy test data only). Prediction quality is carefully assessed through different statistical evaluation metrics and through the assessment of the FitzHugh–Nagumo model solutions calculated for the predicted model parameters.

CNN architectures mostly show the lowest errors when recovering parameters, which can be attributed to their locally acting kernels being advantageous for time-evolving data. Additionally, CNNs extract crucial properties or dynamics of the ODE output when predicting parameters from arbitrarily chosen partial observations; dense NNs, in contrast, perform significantly worse. In only one case can dense networks achieve better results than CNNs, which is when training is performed with noise-free data whereas prediction data are polluted with noise. Thus, dense NNs demonstrate better generalization properties to testing data with "unseen" noise levels. Our NN-based parameter estimation framework can generalize to other time dependent processes, because we successfully carry out a joint estimation of parameters from an ODE model and an autocorrelated noise model.

Future directions include embedding uncertainty quantification methods in NN-based parameter estimations. Uncertainty can be considered with respect to model and data discrepancies but also with respect to network architecture and trained NN parameters. Some recent studies focus on loss functions and propose statistical losses that incorporate additional information about the data, such as an correlation structure (Constantinescu et al., 2020). A different approach is to propose NN architectures for automatically selecting loss functions. Åkesson et al. (2020) shows that loss functions for temporal statistical processes computed from a NN can effectively be used for inferring parameters within an ABC framework. The present work makes use only of the mean squared error as a loss; therefore, one can explore the incorporation of prior information and a description of uncertainty in order to approximate posteriors of the inverse problem. Our reconstruction maps rely on training data generation by solving the forward problem numerous times. To mitigate this computational cost, one can explore reduced order model techniques, which recently have been advanced through machine learning methods (Qian et al., 2020; O'Leary-Roseberry et al., 2020).

Acknowledgments

The effort of Johann Rudi is based in part upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357. The effort of Amanda Lenzi and Julie Bessac is based in part on work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11347.

References

Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017. doi: 10.1088/1361-6420/aa9581.

Mattias Åkesson, Prashant Singh, Fredrik Wrede, and Andreas Hellander. Convolutional neural networks as summary statistics for approximate Bayesian computation. *arXiv preprint arXiv:2001.11760*, 2020.

Leandro M Alonso and Eve Marder. Visualization of currents in neural models with similar behavior and different conductance densities. *eLife*, 8:e42722, 2019. doi: 10.7554/eLife.42722.

Andrea Arnold and Alun L Lloyd. An approach to periodic, time-varying parameter estimation using nonlinear filtering. *Inverse Problems*, 34(10):105005, 2018. doi: 10.1088/1361-6420/aad3e0.

Benjamin Ballnus, Sabine Hug, Kathrin Hatz, Linus Görlitz, Jan Hasenauer, and Fabian J Theis. Comprehensive benchmarking of Markov chain Monte Carlo methods for dynamical systems. *BMC Systems Biology*, 11(64), 2017. doi: 10.1186/s12918-017-0433-1.

Laure Buhry, Sylvain Saighi, Audrey Giremus, Eric Grivel, and Sylvie Renaud. Parameter estimation of the Hodgkin–Huxley model using metaheuristics: application to neuromimetic analog integrated circuits. In *2008 IEEE Biomedical Circuits and Systems Conference*, pages 173–176. IEEE, 2008. doi: 10.1109/BIOCAS.2008.4696902.

Daniela Calvetti and Erkki Somersalo. *Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing*, volume 2 of *Surveys and Tutorials in the Applied Mathematical Sciences*. Springer-Verlag New York, 2007. doi: 10.1007/978-0-387-73394-4.

Ki H Chon and Richard J Cohen. Linear and nonlinear ARMA model parameter estimation using an artificial neural network. *IEEE transactions on biomedical engineering*, 44(3):168–174, 1997.

Emil M Constantinescu, Noémi Petra, Julie Bessac, and Cosmin G Petra. Statistical treatment of inverse problems constrained by differential equations-based models with stochastic terms. *SIAM/ASA Journal on Uncertainty Quantification*, 8(1):170–197, 2020.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020. doi: 10.1073/pnas.1912789117.

Michael Creel. Neural nets for indirect inference. *Econometrics and Statistics*, 2:36–49, 2017.

Noel Cressie and Christopher K Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2015.

Aidan C Daly, David J Gavaghan, Chris Holmes, and Jonathan Cooper. Hodgkin–Huxley revisited: reparametrization and identifiability analysis of the classic action potential model with approximate Bayesian methods. *Royal Society open science*, 2(12):150499, 2015. doi: 10.1098/rsos.150499.

Aidan C Daly, David Gavaghan, Jonathan Cooper, and Simon Tavener. Inference-based assessment of parameter identifiability in nonlinear biological models. *Journal of The Royal Society Interface*, 15(144):20180318, 2018. doi: 10.1098/rsif.2018.0318.

Bin Deng, Jiang Wang, and Yenqiu Che. A combined method to estimate parameters of neuron from a heavily noise-corrupted time series of active potential. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 19(1):015105, 2009. doi: 10.1063/1.3092907.

Shinji Doi, Yuichi Onoda, and Sadatoshi Kumagai. Parameter estimation of various Hodgkin–Huxley-type neuronal models using a gradient-descent learning method. In *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, volume 3, pages 1685–1688. IEEE, 2002. doi: 10.1109/SICE.2002.1196569.

Resat Ozgur Doruk and Laila Abosharb. Estimating the parameters of FitzHugh–Nagumo neurons from neural spiking data. *Brain Sciences*, 9(12):364, 2019. doi: 10.3390/brainsci9120364.

Vivek Dua. An artificial neural network approximation based decomposition approach for parameter estimation of system of ordinary differential equations. *Computers & chemical engineering*, 35 (3):545–553, 2011.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. doi: 10.1016/j.neunet.2017.12.012.

Yuwei Fan, Cindy Orozco Bohorquez, and Lexing Ying. BCR-Net: A neural network based on the nonstandard wavelet form. *Journal of Computational Physics*, 384:1–15, 2019. doi: 10.1016/j.jcp.2019.02.002.

Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445–466, 1961.

Pedro J Gonçalves, Jan-Matthis Lueckmann, Michael Deistler, Marcel Nonnenmacher, Kaan Öcal, Giacomo Bassetto, Chaitanya Chintaluri, William F Podlaski, Sara A Haddad, Tim P Vogels, David S Greenberg, and Jakob H Macke. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, 9:e56261, 2020. doi: 10.7554/eLife.56261.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Ryan N Gutenkunst, Joshua J Waterfall, Fergal P Casey, Kevin S Brown, Christopher R Myers, and James P Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLOS Computational Biology*, 3(10):e189, 2007. doi: 10.1371/journal.pcbi.0030189.

Franz Hamilton, Tyrus Berry, and Timothy Sauer. Tracking intracellular dynamics through extracellular measurements. *PloS one*, 13(10):e0205031, 2018. doi: 10.1371/journal.pone.0205031.

Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the selection of initialization and activation function for deep neural networks. *arXiv preprint arXiv:1805.08266*, 2018.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

Bai Jiang, Tung-yu Wu, Charles Zheng, and Wing H. Wong. Learning summary statistic for approximate Bayesian computation via deep neural network. *Statistica Sinica*, pages 1595–1618, 2017.

Renaud Jolivet, Alexander Rauch, Hans-Rudolf Lüscher, and Wulfram Gerstner. Predicting spike timing of neocortical pyramidal neurons by simple threshold models. *Journal of Computational Neuroscience*, 21(1):35–49, 2006. doi: 10.1007/s10827-006-7074-5.

Jari Kaipio and Erkki Somersalo. *Statistical and Computational Inverse Problems*, volume 160 of *Applied Mathematical Sciences*. Springer-Verlag New York, 2005. doi: 10.1007/b138659.

Yuehaw Khoo and Lexing Ying. SwitchNet: A neural network model for forward and inverse scattering problems. *SIAM Journal on Scientific Computing*, 41(5):A3182–A3201, 2019. doi: 10.1137/18M1222399.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representation*, 2015.

Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

Terence C Mills. *Time Series Techniques for Economists*. Cambridge University Press, 1991.

Jahangir Morshed and Jagath J. Kaluarachchi. Parameter estimation using artificial neural network and genetic algorithm for free-product migration and recovery. *Water Resources Research*, 34(5): 1101–1113, 1998.

Jinichi Nagumo, Suguru Arimoto, and Shuji Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962.

Richard Naud, Brice Bathellier, and Wulfram Gerstner. Spike-timing prediction in cortical neurons with active dendrites. *Frontiers in Computational Neuroscience*, 8:90, 2014. doi: 10.3389/fncom.2014.00090.

Thomas O'Leary-Roseberry, Umberto Villa, Peng Chen, and Omar Ghattas. Derivative-informed projected neural networks for high-dimensional parametric maps governed by PDEs. *arXiv preprint arXiv:2011.15110*, 2020.

Stefano Pagani, Andrea Manzoni, and Alfio Quarteroni. Efficient state/parameter estimation in nonlinear unsteady PDEs by a reduced basis ensemble Kalman filter. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):890–921, 2017. doi: 10.1137/16M1078598.

Jaimit Parikh, James Kozloski, and Viatcheslav Gurev. Integration of AI and mechanistic modeling in generative adversarial networks for stochastic inverse problems. *arXiv preprint arXiv:2009.08267*, 2020.

Astrid A Prinz, Dirk Bucher, and Eve Marder. Similar network activity from disparate circuit parameters. *Nature Neuroscience*, 7(12):1345–1352, 2004. doi: 0.1038/nn1352.

Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, and Karen Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020. doi: https://doi.org/10.1016/j.physd.2020.132401.

Stefan T. Radev, Ulf K. Mertens, Andreass Voss, Lynton Ardizzone, and Ullrich Köthe. BayesFlow: Learning complex stochastic models with invertible neural networks. *arXiv preprint arXiv:2003.06281*, 2020.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Andrew M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, 2010. doi: 10.1017/S0962492910000061.

Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, Philadelphia, PA, 2005.

Karl E Taylor. Summarizing multiple aspects of model performance in a single diagram. *Journal of Geophysical Research: Atmospheres*, 106(D7):7183–7192, 2001.

Werner Van Geit, Erik De Schutter, and Pablo Achard. Automated neuron model optimization techniques: a review. *Biological Cybernetics*, 99:241–251, 2008. doi: 10.1007/s00422-008-0257-6.

Appendix A. Influence of parameters on FitzHugh–Nagumo model solutions and distribution of parameter samples

This section provides background information on how the inference parameters $\boldsymbol{\theta} := (\theta_0, \theta_1)$ affect solutions of the FitzHugh–Nagumo ODE and thus the observational data that is used to generate the NN-based reconstruction maps. Furthermore, the section discusses the distribution of parameter samples that are used to train the networks and test their predictions.

Influence of parameters on FitzHugh–Nagumo model solutions The choice of parameters $\boldsymbol{\theta} := (\theta_0, \theta_1)$ for inference from the FitzHugh–Nagumo model is motivated by how $\boldsymbol{\theta}$ influences solutions of the ODE (1), specifically the membrane potential $u(t)$. Two important characteristics of the oscillating membrane potential are the spike rate and spike duration. Therefore, we visualize these quantities in Figure 5. In order to detect a spike of $u(t)$, we consider a value called spike threshold u_{spike}, which is a constant that determines the occurrence of a spike at time t_{spike}, if $u_{\text{spike}} \leq u(t_{\text{spike}})$ and $u_{\text{spike}} > u(t_{\text{spike}} - \epsilon)$ for some $\epsilon > 0$. The spike duration is defined as the time from t_{spike} until $u_{\text{spike}} > u(t)$ for $t > t_{\text{spike}}$. To generate the plots in Figure 5, we calculate the spike rate (i.e., number of spikes divided by time series length), which is depicted left in Figure 5, and take the average duration over all spike durations to obtain the right plot in Figure 5. We observe in this figure that both the spike rate and the duration are controlled by θ_0 and θ_1 in nonlinear ways, and that the dependencies of spike rate and duration on the parameters are distinct from one another. Additionally, we observe that some combinations of parameters generate zero or only one spike (flat purple, yellow, and white triangular regions in Figure 5). The NN-based reconstruction maps, however, have shown to predict parameters even from these more challenging time series.

Distribution of parameter samples for training and testing of neural networks Training and testing data for NNs are obtained from, first, sampling parameters $\boldsymbol{\theta}$ from their prior distributions (4) and discarding samples outside of prior bounds (5) and, second, solving ODE (1) and storing the membrane potential $u_{\boldsymbol{\theta}}$. Figure 6 presents the distribution of prior samples to illustrate how close training and testing samples of $\boldsymbol{\theta}$ are to one another. The testing samples of size $M = 2000$ are displayed as blue dots and are fixed in all four plots. Fixing the testing samples ensures that the evaluation of prediction errors is comparable across all numerical experiments. The training samples are varied in size $N = 500, 1000, 4000, 8000$ and are shown as black dots in Figure 6 (clockwise from top left scatter plot). The configurations with $N = 500, 1000$ training samples are further apart from each other, hence permitting blue dotted testing samples to be distinct from training sets. The configurations with $N = 4000, 8000$, on the other hand, exhibit dense training sets and cover most of the parameter space.

Appendix B. K-fold cross-validation of selected neural network architectures and their sensitivity to the initialization of weights

We assess further the quality of the selected NN along with the effects of different initial initializations of the weights in fitted dense NNs and CNNs using a k-fold cross-validation, with $k = 6$. After splitting the data set into six roughly equal-sized parts, the NNs are fit to five splits concatenated into a single training set and evaluation metrics are calculated when predicting the remaining sixth part of the data. We repeat this procedure to each of the six data splits, and a final mean and standard deviation of the prediction error are obtained by combining the estimates from the six splits that have been left out from training sets. A 5- to 10-fold cross-validation is typically recommended (Kohavi et al., 1995), and we find that six folds provide a good compromise between bias and variance for our case study.

Tables 12 and 13 show the mean and standard deviation from the 6-fold cross-validation of the prediction skill scores outlined in Section 2.3. Each row of a table corresponds to a different seed when initializing the weights of the dense NNs and CNNs. We notice that the random seeds have little effect on the predictions, indicating that the method is robust to the randomness incurred in the fit of the networks. Moreover, the lower scores from the CNN (Table 13) compared to the dense NN

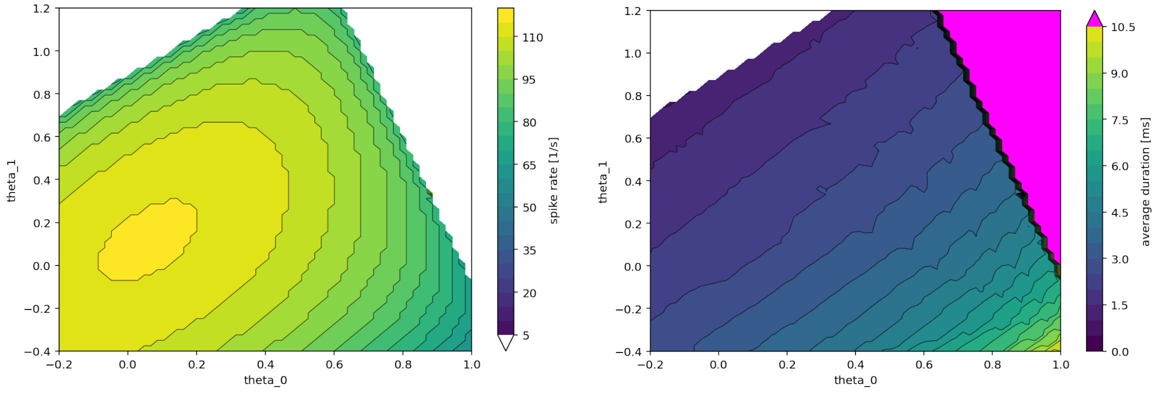

Figure 5: Left: Spike rate of the membrane potential $u(t)$ stemming from solutions of the FitzHugh–Nagumo ODE for varying parameters θ_0 and θ_1, which are on the horizontal and vertical axes, respectively. Right: Average duration of spikes of the membrane potential. The triangular region on the top right in both plots (white color in left, magenta color in right plot) has only a single spike and $u(t)$ stays flat above the spike threshold of 1.5. The triangular region on the top left (white color in both plots), on the other hand, has zero spikes.

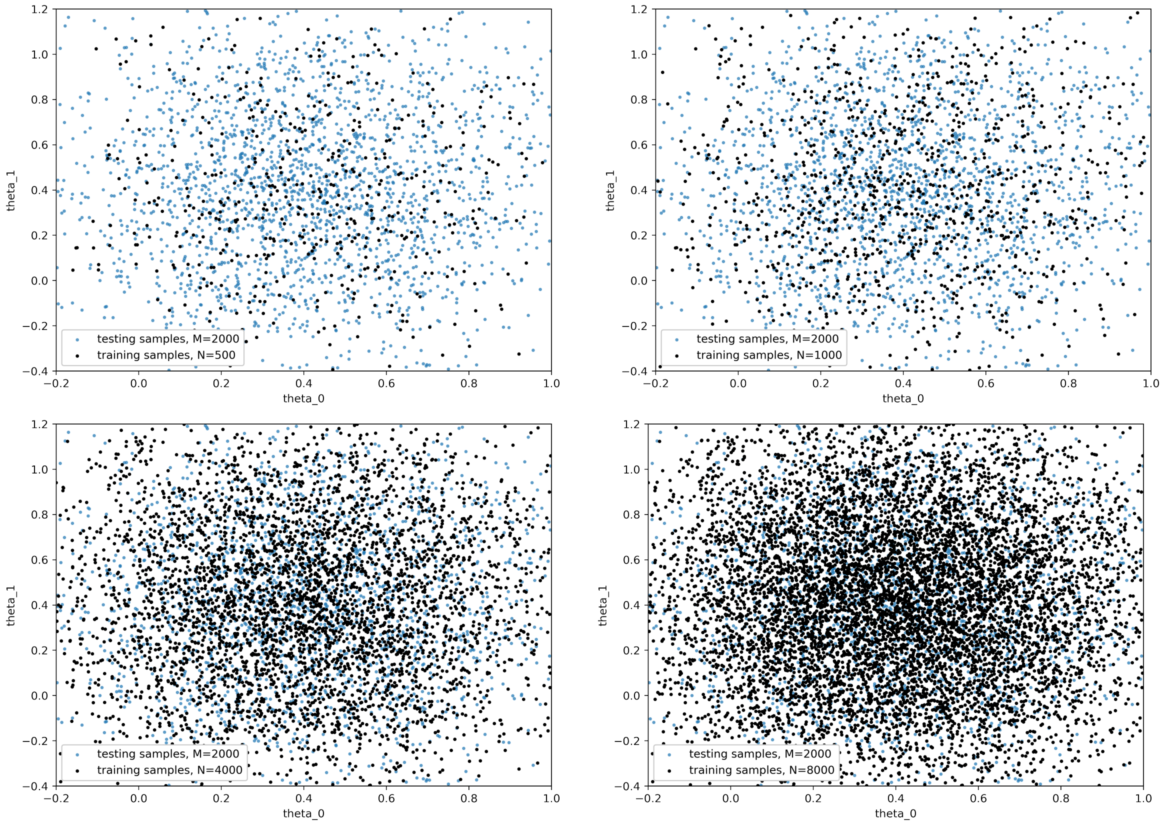

Figure 6: Model parameters of testing set (blue dots) stay fixed ($M = 2000$) while the size of the training set (black dots) increases ($N = 500, 1000, 4000, 8000$), clockwise from the top left scatter plot.

Table 12: 6-fold cross-validation with (noise-free) training and testing sets of sizes $N = 1000$ and $M = 200$, respectively; random seeds for initializing network weights vary per row; using a **dense NN** with 4 layers, $n_u = 32$.

Random seed	**Squared bias**		**C-MSE**		**Median-APE**		**R^2**	
	mean	std.	mean	std.	mean	std.	mean	std.
seed #1	6.04×10^{-5}	5.45×10^{-5}	2.95×10^{-3}	1.13×10^{-3}	0.0350	0.0081	0.970	0.0105
seed #2	5.61×10^{-5}	6.28×10^{-5}	2.45×10^{-3}	9.87×10^{-4}	0.0269	0.0104	0.976	0.0080
seed #3	3.78×10^{-5}	2.40×10^{-5}	2.40×10^{-3}	8.88×10^{-4}	0.0263	0.0063	0.976	0.0075
seed #4	8.11×10^{-5}	5.81×10^{-5}	2.72×10^{-3}	1.25×10^{-3}	0.0281	0.0046	0.973	0.0101
seed #5	5.34×10^{-5}	4.32×10^{-5}	2.31×10^{-3}	4.98×10^{-4}	0.0282	0.0055	0.977	0.0042
seed #6	9.76×10^{-5}	1.49×10^{-4}	2.35×10^{-3}	6.27×10^{-4}	0.0314	0.0047	0.976	0.0053
seed #7	5.55×10^{-5}	7.30×10^{-5}	2.35×10^{-3}	8.62×10^{-4}	0.0271	0.0069	0.977	0.0066
seed #8	9.44×10^{-5}	7.91×10^{-5}	2.52×10^{-3}	6.28×10^{-4}	0.0329	0.0102	0.974	0.0047
seed #9	3.81×10^{-5}	4.21×10^{-5}	2.17×10^{-3}	7.98×10^{-4}	0.0269	0.0053	0.978	0.0063
seed #10	5.09×10^{-5}	4.53×10^{-5}	2.49×10^{-3}	6.41×10^{-4}	0.0295	0.0059	0.975	0.0037

Table 13: 6-fold cross-validation with (noise-free) training and testing sets of sizes $N = 1000$ and $M = 200$, respectively; random seeds for initializing network weights vary per row; using a **CNN** with $n_f \times [1, 2, 4]$, $n_f = 8$.

Random seed	**Squared bias**		**C-MSE**		**Median-APE**		**R^2**	
	mean	std.	mean	std.	mean	std.	mean	std.
seed #1	3.05×10^{-5}	3.05×10^{-5}	5.40×10^{-4}	3.02×10^{-4}	0.0181	0.0038	0.994	0.0032
seed #2	3.59×10^{-5}	4.00×10^{-5}	4.88×10^{-4}	2.60×10^{-4}	0.0166	0.0078	0.994	0.0034
seed #3	2.87×10^{-5}	3.59×10^{-5}	4.48×10^{-4}	2.50×10^{-4}	0.0131	0.0065	0.995	0.0029
seed #4	5.43×10^{-5}	4.10×10^{-5}	4.60×10^{-4}	2.11×10^{-4}	0.0176	0.0054	0.994	0.0025
seed #5	3.60×10^{-5}	3.58×10^{-5}	5.40×10^{-4}	3.21×10^{-4}	0.0191	0.0036	0.994	0.0036
seed #6	2.24×10^{-5}	2.74×10^{-5}	5.01×10^{-4}	2.77×10^{-4}	0.0169	0.0066	0.994	0.0034
seed #7	4.77×10^{-5}	3.85×10^{-5}	4.92×10^{-4}	2.77×10^{-4}	0.0167	0.0047	0.994	0.0034
seed #8	2.83×10^{-5}	2.71×10^{-5}	5.12×10^{-4}	1.77×10^{-4}	0.0140	0.0058	0.994	0.0022
seed #9	1.13×10^{-5}	9.92×10^{-6}	4.47×10^{-4}	2.67×10^{-4}	0.0961	0.0035	0.995	0.0032
seed #10	2.34×10^{-5}	1.35×10^{-5}	4.84×10^{-4}	2.62×10^{-4}	0.0186	0.0062	0.994	0.0031

(Table 12) substances the conclusions in Section 3.2 that the CNN results in better predictions than the dense NN.

Appendix C. Extended results for simulation of FitzHugh–Nagumo model from predicted parameters

We support the observations from the last paragraph of Section 3.4 here by showing a wider scope of prediction errors. Recall that Section 3.4 considers the evaluation of parameter prediction errors from neural networks by means of comparing simulations of the FitzHugh–Nagumo model from predicted parameters with the time series used as input to the NN. The predictions are carried out by the CNN selected in Section 3.2 with $n_f \times [1, 2, 4]$, $n_f = 8$.

We show the following three Figures. Figure 7 represents the case where both training and testing data do not contain noise; in Figure 8 only testing data contain noise; and Figure 9 shows results where both training and testing data contain noise. Each figure contains five graphs where each graph is selected from the testing data set (of size $M = 2000$) as a specific quantile of the MSE, in order to show the effects of a range of prediction accuracies on the FitzHugh–Nagumo model solutions. We present the 10th, 25th, 50th (i.e., median), 75th, and 90th percentiles of MSE between simulated and testing time series'. In Figure 7, we observe a nearly optimal overlapping of simulated output and test data for the noise-free case, even for the 90th percentile of MSE. When noise is added to training or testing data (Figures 8 and 9), the simulated time series show shifted spiking frequencies, which become increasingly pronounced for the median, 75th, and 90th percentiles. By comparing the predicted $\boldsymbol{\theta} = (\theta_0, \theta_1)$ with the corresponding test values (given in graph labels in Figure 8), we observe that the prediction accuracy for θ_1 degrades with increasing percentile. This can imply that a larger contribution to discrepancies of the simulated ODE output stems from inaccurate predictions of parameter θ_1. Overall, the better correspondence of the time series' in Figure 9 compared with Figure 8 demonstrates the significant benefit of using noisy training data as it is done in the former figure. This observation supports the results in Section 3.4.

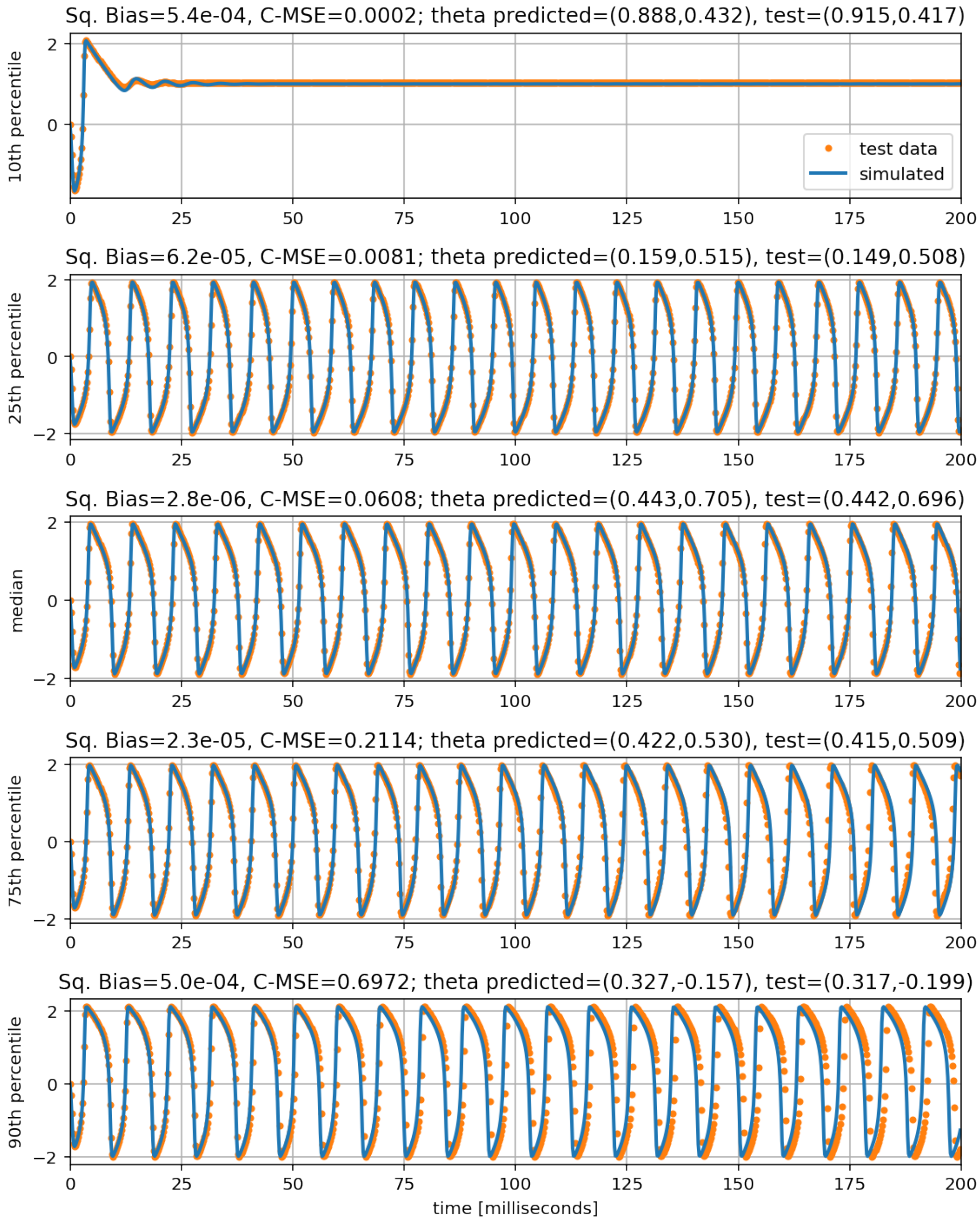

Figure 7: Simulations of FitzHugh–Nagumo model (blue lines) using parameters from CNN predictions; corresponding data that gave rise to prediction are shown as orange dots. Training samples ($N = 1000$) and testing samples ($M = 2000$) are both noise-free. Graphs from top to bottom show increasing quantiles of MSE between true and estimated time series.

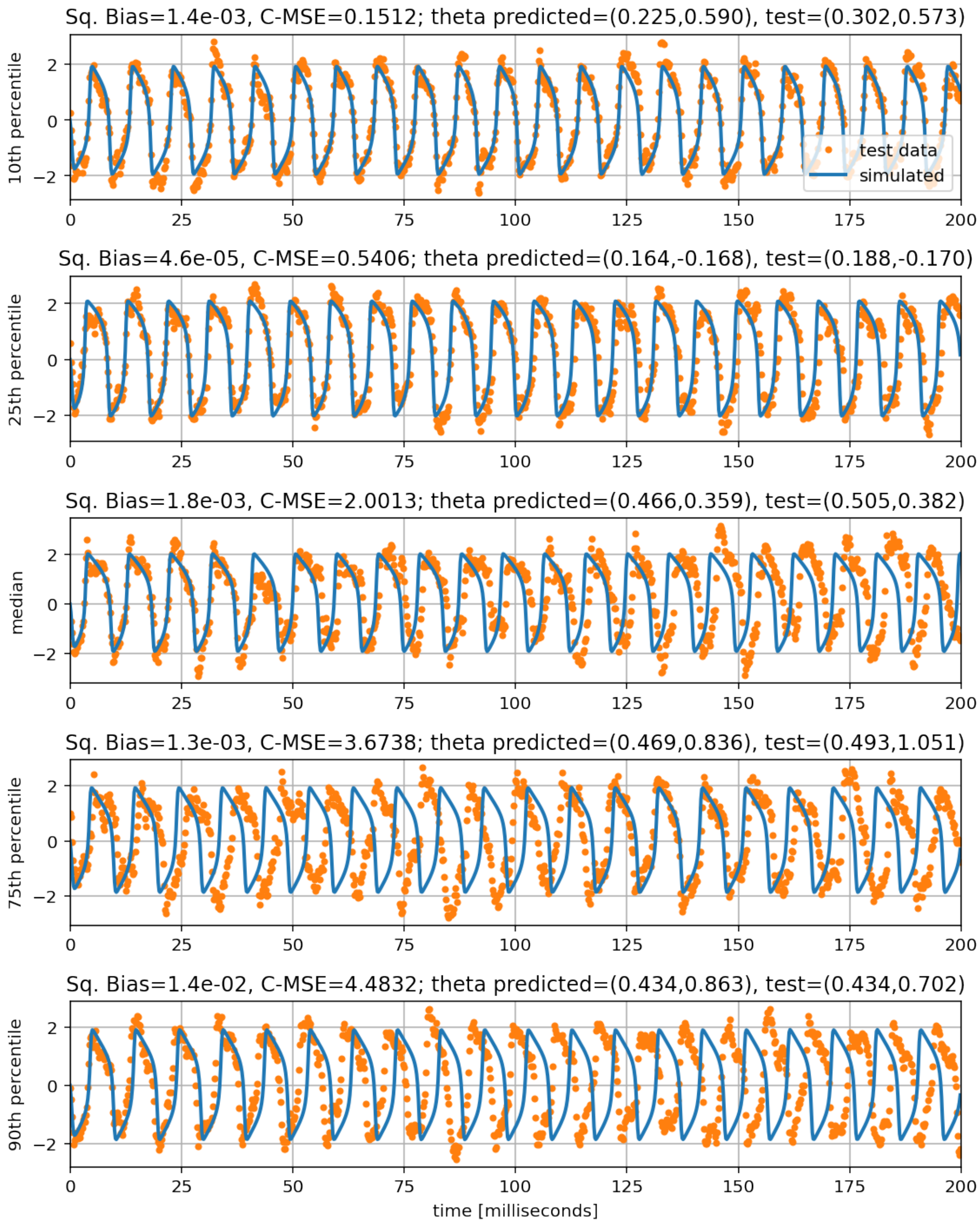

Figure 8: Simulations of FitzHugh–Nagumo model (blue lines) using parameters from CNN predictions; corresponding data that gave rise to prediction are shown as orange dots. Training samples ($N = 1000$) are noise-free but testing data ($M = 2000$) contains noise. Graphs from top to bottom show increasing quantiles of MSE between true and estimated time series.

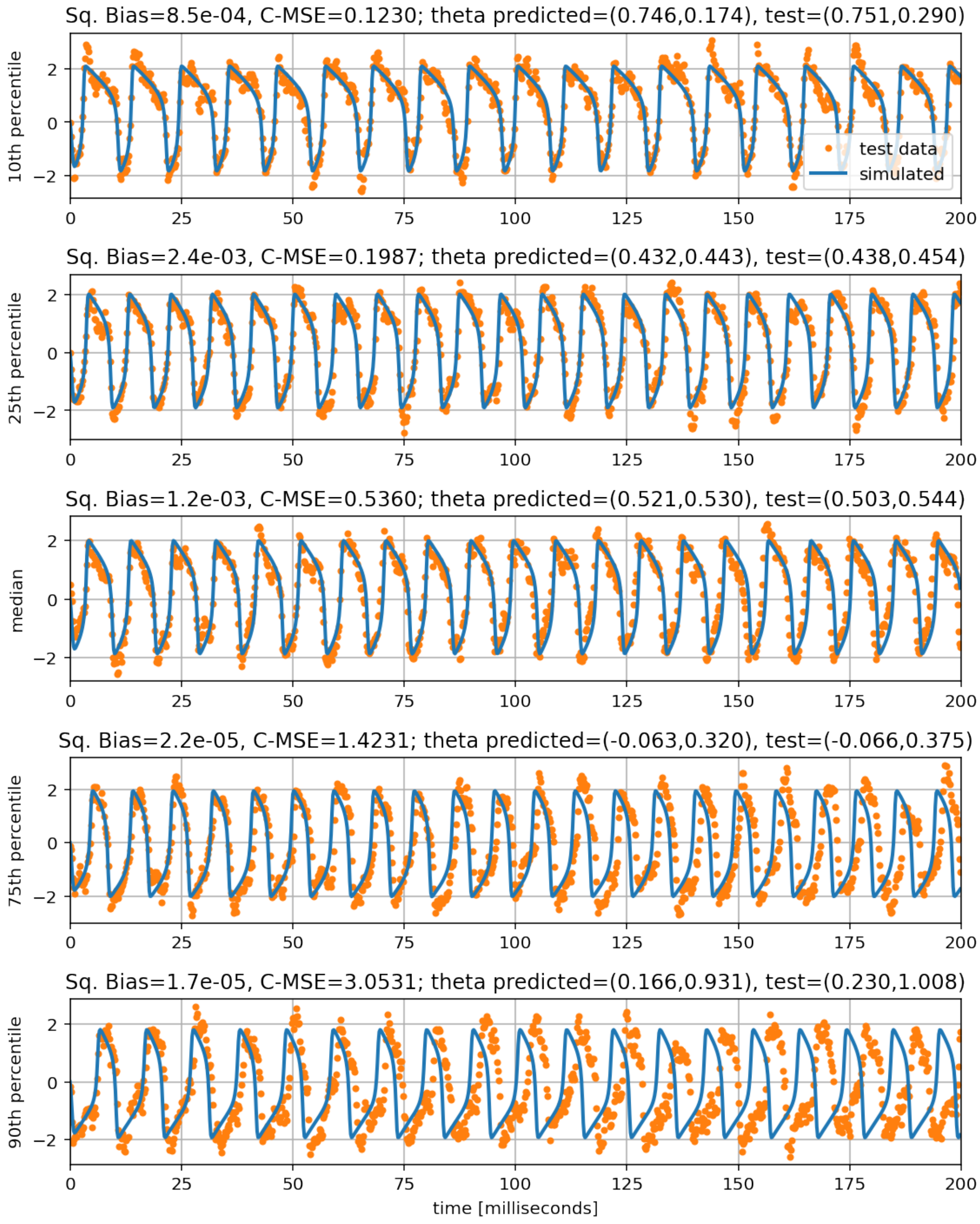

Figure 9: Simulations of FitzHugh–Nagumo model (blue lines) using parameters from CNN predictions; corresponding data that gave rise to prediction are shown as orange dots. Training samples ($N = 1000$) and testing samples ($M = 2000$) both contain noise. Graphs from top to bottom show increasing quantiles of MSE between true and estimated time series.