8th International Conference on Computational and Mathematical Biomedical Engineering - CMBE2024
24–26 June 2024, USA
P. Nithiarasu and R. Löhner (Eds.)

DATA REPRESENTATIONS FOR PARAMETER ESTIMATION WITH DEEP LEARNING MODELS FOR A DYNAMICAL SYSTEM

Johann Rudi[1], **German Villalobos**[2,3], **and Andreas Mang**[2]
[1]Virginia Tech, Blacksburg, Virginia, jrudi@vt.edu
[2]University of Houston, Houston, Texas, amang@central.uh.edu
[3]Layher, Inc., Houston, Texas

SUMMARY

Dynamical systems are ubiquitous in scientific applications and these are often governed by parametrized equations that are deterministic differential equations and/or stochastic processes. We developed parameter estimation techniques based on deep learning, where we train (artificial) neural networks to estimate parameters of such systems. Advantages of our approach are fast parameter predictions, which, as a consequence, accelerate applications with real-time and frequent estimation demands. Furthermore, the method does not require a data misfit or likelihood term to be prescribed in the definition of the inverse problem. In this presentation, we summarize how neural networks are learning inverse maps, and we investigate the effects of different representations of data from a dynamical system's output on the estimation accuracy of the system's parameters, where the data is used to train a neural network.

Key words: *deep learning, neural networks, inverse problems, parameter estimation, dynamical systems, FitzHugh–Nagumo*

1 INTRODUCTION

Generally, we address the scientific problem of computationally learning neural network-based inverse maps for parameter estimation in scientific models. We consider inverse problems that aim to estimate parameters from given observational data. The parameters give rise to the data through the solution of a parametrized physical or statistical model. Such inverse problems have in the past been solved deterministically with techniques from optimization as well as in a statistical/Bayesian framework using sampling-based methods. These approaches exhibit computational challenges prohibiting their effective use for certain scientific applications. We propose to computationally learn solution operators for inverse problems based on (artificial) neural networks (NNs), which we introduced in [1].

Our goal is to construct artificial NNs to solve inverse problems, where we seek to estimate parameters of a deterministic dynamical system, which is introduced below. We train NNs to approximate an *inverse map (IM)* that estimates parameters of a dynamical system from observational data, where the observational data are governed by the solutions of the dynamical system. While dynamical systems arise frequently in many applications, we focus on dynamics of biological neurons.

We consider the FitzHugh–Nagumo [2, 3] system of ordinary differential equation (ODE). This ODE system describes spiking neurons via a two equations,

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

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

where the unknowns of the ODE are the membrane potential $u = u(t)$ and the recovery variable $v = v(t)$. $\zeta = -0.4$ denotes the total membrane current and is a stimulus applied to the neuron, which we assume to be constant in time. $\gamma = 3.0$ determines the strength of damping and is assumed

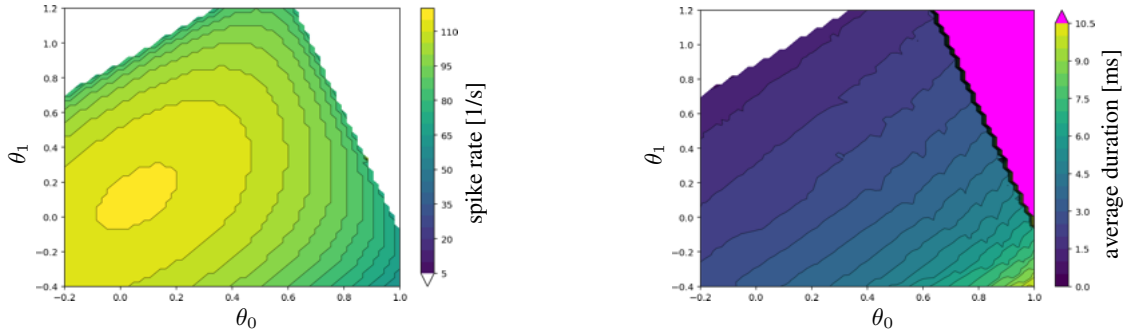

Figure 1: *Left:* Spike rate of the membrane potential $u(t)$ stemming from solutions of (1) for varying parameters θ_0 and θ_1 (on horizontal and vertical axes, resp.). *Right:* Average duration of spikes of the membrane potential (i.e., duration of the potential above a threshold). The triangular region 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.

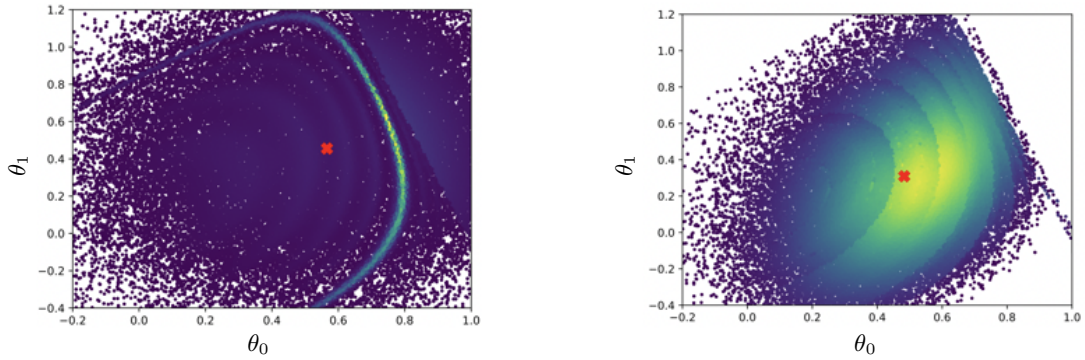

Figure 2: Demonstrating the challenges of parameter estimation: We show samples of the posterior of one inverse problem, associated with one true θ^*. The posterior was obtained by 100,000 samples of a Markov-chain Monte Carlo method, which is relatively computationally costly for a single inference. The prior mean is indicated with a *red cross*. *Left:* Posterior distribution with time series data. *Right:* Posterior distribution with spike feature data (rate and duration).

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 Fig. 1). ODE (1) is augmented with initial conditions, in our case $u(0) = v(0) = 0$.

In [1], we addressed significant computational challenges, which certain time-evolving models pose in an inference setting. For the inverse problem, where we are given observational data $d(t) := u_{\boldsymbol{\theta}^*}(t) + \eta(t)$, composed of $u_{\boldsymbol{\theta}^*}(t)$, which is the first component (1a) of the solution of ODE (1) with unknown true parameters $\boldsymbol{\theta}^* := (\theta_0^*, \theta_1^*)$, and added correlated noise $\eta(t)$. Our goal is to find parameters $\boldsymbol{\theta} := (\theta_0, \theta_1)$ that are consistent with data $d(t)$ and model output $u_{\boldsymbol{\theta}}(t)$ for all t. In a Bayesian framework, 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 $\pi(\boldsymbol{\theta})$. We write the negative log of the posterior as

$$-\log\Big(\pi(\boldsymbol{d} \,|\, \boldsymbol{\theta})\,\pi(\boldsymbol{\theta})\Big) = \left\| \frac{d(t) - u_{\boldsymbol{\theta}}(t)}{\sigma_{\text{noise}}} \right\|_{L_2}^2 + \tfrac{1}{2}\,\big|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}_{\text{prior}}\big|_{\Sigma_{\text{prior}}^{-1}}^2, \tag{2}$$

where σ_{noise} denotes the standard deviation of the data noise, which is related to η in $d(t)$; $\bar{\boldsymbol{\theta}}_{\text{prior}} := (\bar{\theta}_{0,\text{prior}}, \bar{\theta}_{1,\text{prior}})$ is the prior mean, and $\Sigma_{\text{prior}} := \text{diag}(\sigma_{0,\text{prior}}, \sigma_{1,\text{prior}})$ is the prior standard deviation. We overcome the highly nonlinear and nonconvex (negative log of the) posterior (2), shown in Fig. 2, left, of an inverse problem with a ODE (1) as the model, and successfully address the challenge that prior information about parameters is very limited. Additionally, we recover parameters from an autocorrelated statistical noise model that is polluting the ODE's outputs (which is not further discussed in this presentation). These challenges can cause traditional optimization to fail and alternative algorithms to exhibit large computational costs. Our results demonstrate that NNs have the potential to

estimate parameters in dynamical models and stochastic processes, and they are capable of predicting parameters accurately. Furthermore, trained NNs can compute estimations instantly, thus enabling real-time inference applications.

2 METHODOLOGY

Neural network model architecture The NN architecture resembles models used for image classification [4]. The architecture is shown in Fig. 3, where the first sequence of n_{conv} hidden layer groups contain a convolutional, an activation and a pooling layer; it is followed by a sequence of n_{lin} hidden layer groups with a linear (fully connected) and an activation layer. Reshaping takes place in between the two groups to transform outputs of convolutional layers to the inputs required for linear layers. The input data (i.e., features) is time series data or spike feature data, and the outputs (i.e., targets) are parameters $\boldsymbol{\theta}$ of the ODE (1). More details about the NN model architecture can be found in [1].

Data representation We consider the following input data to the NN: either (i) time series data (1000 time steps) or (ii) spike feature data (spike rate and duration). As a result one training sample for the NN has input sizes: either (i) $\boldsymbol{x} \in \mathbb{R}^{1000}$ for time series data or (ii) $\boldsymbol{x} \in \mathbb{R}^2$ for spike feature data. While the data size for spike features is significantly less than the time series, it still captures sensitivities with respect to the parameters $\boldsymbol{\theta}$ that we aim to infer, which is demonstrated in Fig. 1. The output of the NN for both input cases is of size: $\boldsymbol{y} = \boldsymbol{\theta} \in \mathbb{R}^2$. The number of training samples is 2,000 and the number of testing samples is also 2,000.

3 RESULTS AND CONCLUSIONS

We study the capabilities of NNs to predict parameters $\boldsymbol{\theta}$ of the ODE (1), where we contrast different data representations (time series vs. spike features) detailed in Sec. 2. The error in the predicted parameters from a trained NN is shown in Figs. 4 and 5, where the diagonal line represents the ideal (error-free) prediction. In both figures, the respective NN's architectures have been optimized to best handle the inverse map for the given input data type. We see an overall higher accuracy in predictions when the entire time series is used (Fig. 4) compared with spike features (Fig. 5), which were extracted from the time series. The trend of better predictions is not surprising and tells that with sufficient network optimization, a NN is able to reveal additional features that our spike features were not able to capture. However, the NN required to handle time series is significantly more complex and the data sets are larger (here by a factor of 500 larger). This demonstrates the tradeoffs one has to make regarding storage requirements, network complexity, and accuracy of predictions.

While in this abstract only two representations of data were presented, we will investigate other representations, such as, Fourier transformations of time series data and summary representations that are obtained through nonsupervised machine learning methods applied to the time series.

ACKNOWLEDGEMENTS

This work was partly supported by the National Science Foundation (NSF) through the grants DMS-2009923 and DMS-2012825. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] J Rudi, J Bessac, and A Lenzi. "Parameter Estimation with Dense and Convolutional Neural Networks Applied to the FitzHugh–Nagumo ODE". In: *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*. Ed. by J Bruna, J Hesthaven, and L Zdeborova. Vol. 145. Proceedings of Machine Learning Research. PMLR, 2022, pp. 781–808. URL: https://proceedings.mlr.press/v145/rudi22a.html.

[2] R FitzHugh. "Impulses and Physiological States in Theoretical Models of Nerve Membrane". In: *Biophysical Journal* 1.6 (1961), pp. 445–466. DOI: 10.1016/s0006-3495(61)86902-6.

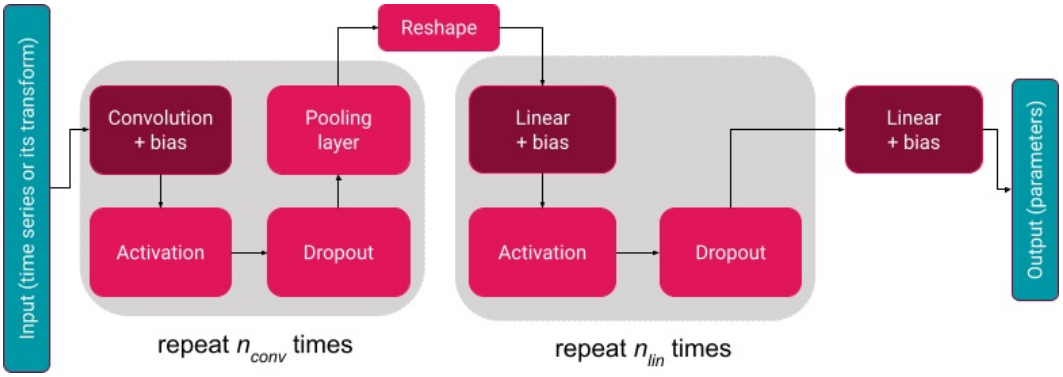

Figure 3: NN model architecture with a sequence of n_{conv} layer groups containing a convolutional, an activation and a pooling layer followed by a sequence of n_{lin} layer groups with a linear (fully connected) and an activation layer. Layers with trainable parameters are shown in *dark red*. The input is time series data or spike features data and the outputs are parameters of an ODE and a statistical process for noise.

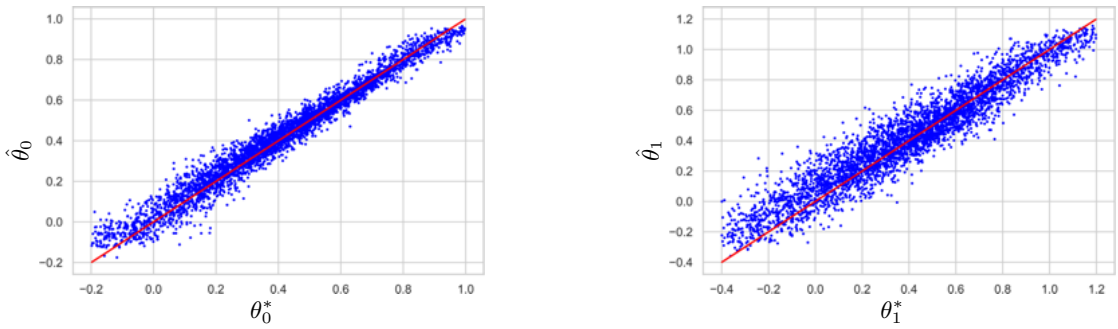

Figure 4: True vs. NN-predicted values of parameters θ_0 *(left)* and θ_1 *(right)*. The time series data was used as the NN's input for training and evaluation. Shown are predictions of testing samples, not presented to NN during the training. The NN's architecture is based on Fig. 3 using $n_{\text{conv}} = 3$ with 8, 16, 32 number of convolution channels, $n_{\text{lin}} = 2$ with 32 linear units. It was selected as optimal for this data after an architecture search.

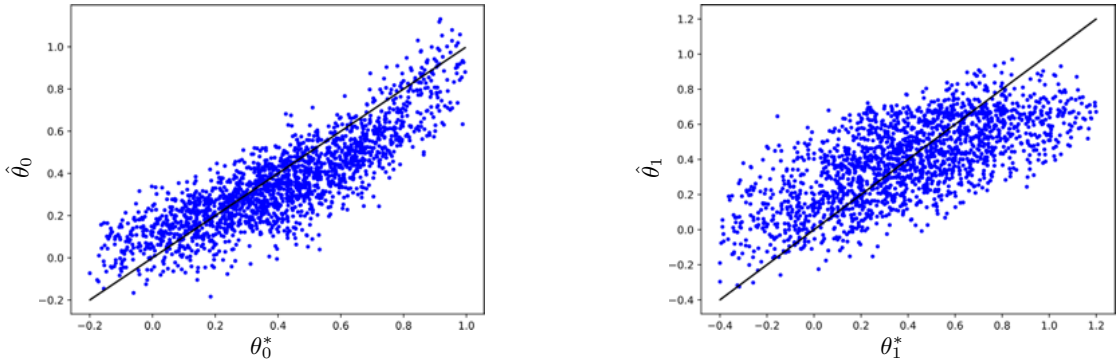

Figure 5: True vs. NN-predicted values of parameters θ_0 *(left)* and θ_1 *(right)*. The spike feature data was used as the NN's input for training and evaluation. Shown are predictions of testing samples, not presented to NN during the training. The NN's architecture is based on Fig. 3 using $n_{\text{conv}} = 0$, $n_{\text{lin}} = 8$ with 16 linear units. It was selected as optimal for this data after an architecture search.

[3] J Nagumo, S Arimoto, and S Yoshizawa. "An Active Pulse Transmission Line Simulating Nerve Axon". In: *Proceedings of the IRE* 50.10 (1962), pp. 2061–2070. DOI: 10.1109/JRPROC.1962.288235.

[4] A Krizhevsky, I Sutskever, and GE Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90. DOI: 10.1145/3065386.