Solving Neural Field Equations using Physics Informed Neural Networks

Weronika Wojtak,[1, 2, 3, a)] Estela Bicho,[1, b)] and Wolfram Erlhagen[2, c)]

[1]*Research Centre Algoritmi, University of Minho, Guimarães, Portugal*
[2]*Research Centre of Mathematics, University of Minho, Guimarães, Portugal*
[3]*Centro de Computação Gráfica, Guimarães, Portugal*

[a)]Corresponding author: w.wojtak@dei.uminho.pt
[b)]estela.bicho@dei.uminho.pt
[c)]wolfram.erlhagen@math.uminho.pt

Abstract This article presents an approach for solving neural field equations (NFEs) using Physics Informed Neural Networks (PINNs). NFEs are integro-differential equations describing the spatio-temporal dynamics of neuronal populations in the cortex. The traditional numerical methods for NFEs require significant computational effort due to the discretization of the spatial convolution. The proposed approach leverages Fast Fourier Transforms (FFTs) to reduce the computational cost and improve efficiency. A PINN, consisting of a surrogate network and a residual network, is trained to approximate the solutions of NFEs. The effectiveness of the approach is demonstrated by solving the one-dimensional Amari equation, a commonly used neural field formulation. Our results show that the accuracy of the PINN approach is comparable to traditional numerical methods. Future research directions include optimizing hyperparameters, incorporating input terms in NFEs, exploring transfer learning, addressing the inverse problem, and extending the approach to higher dimensions.

INTRODUCTION

Neural field equations (NFEs) play an important role in the understanding of neuronal activity. They are typically formalized by integro-differential equations (IDEs) describing the large-scale spatio-temporal dynamics of neuronal populations in the cortex. NFEs have various applications in neuroscience and robotics, for an overview we refer the reader to [1]. The mathematical analysis is usually focused on the existence and stability of localized activation patterns which are believed to represent a basis for cortical information processing. Since in general no closed-form expressions can be found for these patterns, the solutions must be approximated numerically. This requires significant computational effort, mostly due to the discretization of the spatial convolution term. Several effective approaches to solving NFEs numerically have been proposed, including collocation techniques [2], Galerkin schemes [3], low-rank methods [4], or employing quadrature rules [5]. One of the most efficient ways to reduce the computational effort required by numerical integration is the use of Fast Fourier Transforms (FFTs) [6].

Here, we present an approach for solving neural field equations using Physics Informed Neural Networks (PINNs) [7]. PINNs are neural networks designed to solve problems involving Partial Differential Equations (PDEs). Their architecture consists of two parts: a deep learning network called an approximator or a surrogate network, and a residual network encoding the governing PDEs. The surrogate network is trained to provide an approximate solution at a given point in the simulation domain (also called collocation point). The residual network receives the output of the surrogate network and calculates a residual value (also called loss function). The derivatives on the residual network graph are calculated using automatic differentiation (AutoDiff), that is, the chain rule is repeatedly applied, which is the main strength of PINNs [7]. For a recent review of PINNs and their applications, we refer the reader to [8] (also see [9]). There exist several libraries for implementing PINNs, e.g., DeepXDE [10] and sciANN [11].

To illustrate our method, we use a PINN to solve the one-dimensional Amari equation, an IDE widely used in dynamic neural field models [12]. The basic formulation of PINNs for solving IDEs requires the discretization of the integral term in order to transform IDEs into ordinary differential equations (ODEs). This typically causes two problems: the discretization and truncation errors and the high computational cost required to achieve satisfactory accuracy [13]. Here, we mitigate this issue by leveraging the fact that the integral term in the Amari equation is a spatial convolution and as such can be efficiently computed using FFTs [6]. This not only speeds up the calculation of the integral term but also provides the periodic boundary conditions for our problem, eliminating the need to include them in the PINN's loss function. To the best of our knowledge, this is the first attempt to solve a neural field equation by means of neural networks. Our first results demonstrate that the accuracy of this approach is comparable with traditional methods used for solving NFEs numerically, opening a number of new possibilities for future studies.

NEURAL FIELD EQUATIONS

We solve the canonical Amari neural field equation [12] defined on a one-dimensional finite domain Ω

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= -u(x,t) + \int_{\Omega} w(|x-y|) f(u(y,t) - \kappa) \mathrm{d}y, \quad (x,t) \text{ in } \Omega \times M, \\ u(x,0) &= u_0, \quad x \text{ in } \Omega, \end{aligned} \tag{1}$$

where $u(x,t)$ represents the activity of a neuron at the spatial location x at time t. The firing rate function $f(x)$ is chosen as a Heaviside function with threshold κ such that $f(x) = 1$ for $x \geq 0$ and $f(x) = 0$ otherwise. The connectivity function w is chosen as a scaled version of an oscillatory connectivity function [14]

$$w_{osc}(x) = A_{osc} e^{-b|cx|} (b \sin|cx| + \cos(cx)), \tag{2}$$

where A_{osc} controls the amplitude and b controls the rate at which the oscillations decay with distance. The parameter c is added to scale the spatial extend of the kernel. The initial condition u_0 is given by

$$u_0 = u(x,0) = A_0 e^{\left(-x^2/\sigma_0^2\right)}, \tag{3}$$

and the boundary conditions are periodic. Finally, $M = [0,T]$ defines the time interval for the activity integration.

PHYSICS INFORMED NEURAL NETWORKS

Physics informed neural networks (PINNs) can solve differential equations expressed in the form

$$\begin{aligned} \mathcal{F}(u(z);\gamma) &= f(z) \quad z \text{ in } \Omega, \\ \mathcal{B}(u(z)) &= g(z) \quad z \text{ in } \partial\Omega, \end{aligned} \tag{4}$$

defined on the domain $\Omega \subset \mathbb{R}$ with the boundary $\partial\Omega$, where $z = [x_1, \ldots, x_{d-1}; t]$ indicates the space-time coordinate vector, u represents the unknown solution, γ are the parameters of the governing equation, f is the function identifying the data of the problem and $\mathcal{F}$ is the non-linear differential operator [8]. $\mathcal{B}$ denotes the operator indicating arbitrary initial or boundary conditions and g is the boundary function. The initial condition can be thought of as a type of Dirichlet boundary condition on the spatio-temporal domain. Since the integral in (1) represents a spatial convolution, it can be solved numerically using a Fast Fourier Transform (FFT). This imposes periodic boundary conditions in x, we thus drop the boundary condition component in the PINN formulation.

The inputs (x,t) to the neural network are the coordinates of the training points which consist of two parts: the sampling points $(x_{ini}, 0)$ of the initial condition and collocation points (x_f, t_f) in the equation domain.

In PINNs, $u(z)$ is computationally predicted by a neural network, giving rise to an approximation

$$u_{pred}(z;\theta) \approx u(z), \tag{5}$$

where θ represents the weights W and biases b of the neural network. Learning to approximate the solution of the IDE takes place by minimizing a loss function which depends on the governing equation and the initial conditions. A schematic of a PINN for solving the equation (1) is shown in Fig. 1.

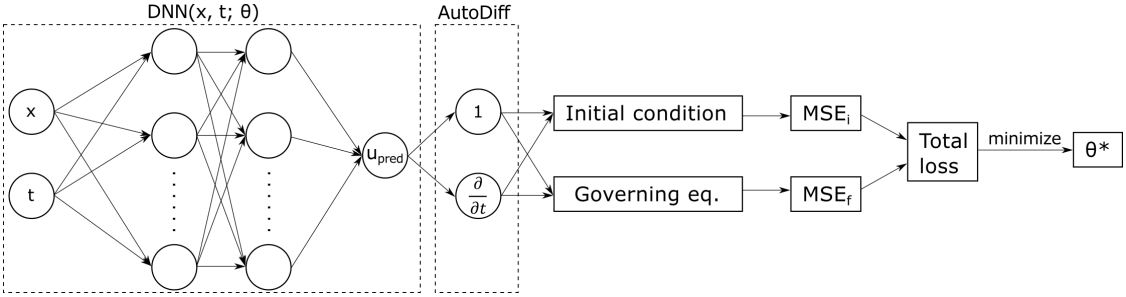

Figure 1. Schematic of a PINN for solving the neural field equation (1).

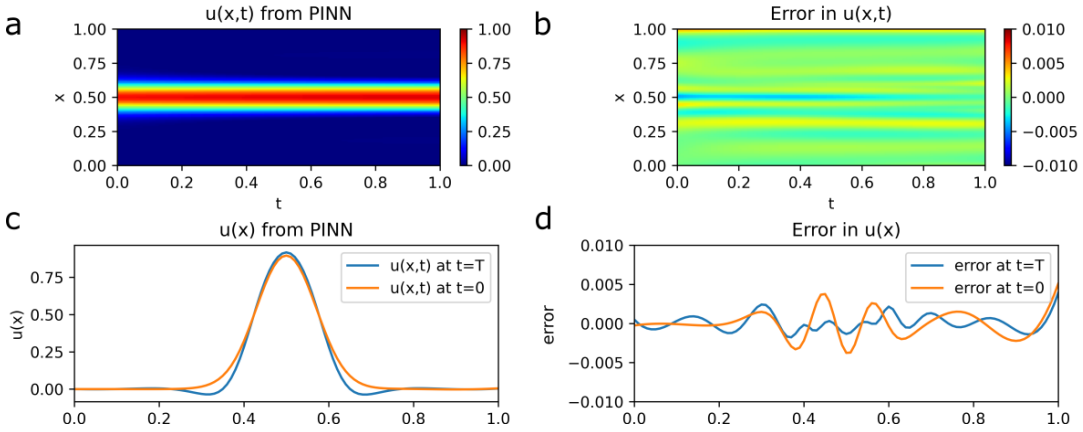

Figure 2. Localized activity pattern or bump solution of the Amari equation. The approximation by the PINN and the respective approximation error (the difference between the PINN solution and the one obtained by the forward Euler method) are shown for the time interval $M = [0,1]$ (a, b) and for snapshots at times $t = 0$ and $t = 1$ (c, d). The initial condition at time $t = 0$ is given by (3) with $A_0 = 0.9$ and $\sigma_0 = 0.07$. The connectivity function is (2) with $A_{osc} = 10$, $b = 0.7$ and $c = 24$. Threshold $\kappa = 0.4$.

RESULTS

The approximate solution to our problem is found by the surrogate network. We approximate a so called *bump* solution as a special case of interest due to the applications. The residual network encodes the Amari equation and provides the loss function to drive the surrogate network's optimization. We used a simple fully connected feed-forward neural network architecture with no additional regularization. Specifically, we employed a PINN with two hidden layers and 40 neurons in each layer. We used the hyperbolic tangent (*tanh*) as the activation function, as it satisfies the smoothness requirements for PINNs [15]. To optimize the loss function, the L-BFGS optimizer with 0.01 learning rate was used. To implement our approach, we adapt the PyTorch code available in [13]. The code and trained model are available at `https://github.com/w-wojtak/solving-NFEs-using-PINNs`.

For the neural field equation (1), the loss function MSE_{total} is determined by

$$MSE_{total} = MSE_i + MSE_f \tag{6}$$

where

$$MSE_i = \frac{1}{N_i}\sum_{i=1}^{N_i} |u_{pred}(x_i^{ini}, 0; \theta) - u_0(x_i^{ini}, 0; \theta)|^2 \tag{7}$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |\frac{\partial u_{pred}(x_i^f, t_i^f; \theta)}{\partial t} + u_{pred}(x_i^f, t_i^f; \theta) - w \otimes H(u_{pred} - \kappa)(x_i^f, t_i^f; \theta)|^2 \tag{8}$$

represent the mean square errors of the residuals of the initial condition and the governing equation, respectively. The symbol $\otimes$ represents the spatial convolution, w is the connectivity function (2) and H is the Heaviside function with threshold κ.

The training data set consists of two parts: 500 initial points $(x_i^{ini}, 0)$ uniformly sampled at $t = 0$ and 10000 collocation points (x_i^f, t_i^f), uniformly sampled in the equation domain. The approximation by the PINN of a localized activity pattern of bump solution after 500 iterations and the error are presented in Fig. 2. The relative L2 error between the PINN solution and the approximation obtained by the forward Euler method is 0.37%.

CONCLUSION AND OUTLOOK

We have shown that the PINN framework can be successfully employed for solving neural field equations. Using FFTs reduces the computational effort needed for computing the integral term in the Amari equation and allows for

using vanilla PINNs without any additional mechanisms such as auxiliary outputs [13]. A comparison of the solutions obtained with the PINN and the numerical forward Euler method frequently used in applications (e.g., [16]) revealed a relative error below one percent. This demonstrates that the PINN framework is suitable for solving NFEs, providing a new method for tackling this type of equations.

The promising first results open new perspectives for further improvements and model extensions. Here, we used a scaled version of the connectivity kernel (2) and a relatively small equation domain, i.e., $[0, 1]$ for both space and time dimensions. This choice was motivated by the use of *tanh* as nonlinear activation function with output values between -1 and +1. This restriction can be mitigated for example by adding a normalization layer to the network. To improve the accuracy of the solution, the hyperparameters of the network (e.g., number of layers and neurons, learning rate, or number of epochs) must be carefully adjusted. This can be done for example using a random search strategy [17]. In the present paper, we considered the Amari equation without external inputs, however from the applications point of view, including the input term in the equation is crucial [3, 16]. We thus plan to extend this work to the case of NFEs with inputs and to leverage the transfer learning that the PINN framework offers [9]. Transfer learning allows for training a PINN with a given input and then using its trained parameters (weights and biases) to initialize networks for solving equations with different inputs, which significantly speeds up the training process. Another compelling possibility is modifying the current approach to tackle the inverse problem, i.e., inferring the NFE parameters from potentially noisy data [8]. It would be also interesting to extend our study to NFEs in two or more spatial dimensions or even to equations posed on surfaces with more complex geometries [2]. These are research directions that we intend to explore in the near future.

ACKNOWLEDGMENTS

The work received financial support by Portuguese Funds through FCT (Fundação para a Ciência e a Tecnologia) within the R&D Projects UIDB/00319/2020 and UIDB/00013/2020.

REFERENCES

1. S. Coombes, P. beim Graben, R. Potthast, and J. Wright, *Neural fields: theory and applications* (Springer, 2014).
2. R. Martin, D. J. Chappell, N. Chuzhanova, and J. J. Crofts, "A numerical simulation of neural fields on curved geometries," Journal of computational neuroscience **45**, 133–145 (2018).
3. P. Lima, W. Erlhagen, M. Kulikova, and G. Y. Kulikov, "Numerical solution of the stochastic neural field equation with applications to working memory," Physica A: Statistical Mechanics and its Applications **596**, 127166 (2022).
4. W.-J. Xie and F.-R. Lin, "A fast numerical solution method for two dimensional Fredholm integral equations of the second kind," Applied Numerical Mathematics **59**, 1709–1719 (2009).
5. G. Faye and O. Faugeras, "Some theoretical and numerical results for delayed neural field equations," Physica D: Nonlinear Phenomena **239**, 561–578 (2010).
6. A. Hutt and N. Rougier, "Activity spread and breathers induced by finite transmission speeds in two-dimensional neural fields," Physical Review E **82**, 055701 (2010).
7. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational physics **378**, 686–707 (2019).
8. S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: where we are and what's next," Journal of Scientific Computing **92**, 88 (2022).
9. S. Markidis, "The old and the new: Can physics-informed deep-learning replace traditional linear solvers?" Frontiers in big Data , 92 (2021).
10. L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," SIAM review **63**, 208–228 (2021).
11. E. Haghighat and R. Juanes, "SciANN: A Keras/Tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks," Computer Methods in Applied Mechanics and Engineering **373**, 113552 (2021).
12. S. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," Biological Cybernetics **27**, 77–87 (1977).
13. L. Yuan, Y.-Q. Ni, X.-Y. Deng, and S. Hao, "A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations," Journal of Computational Physics **462**, 111260 (2022).
14. F. Ferreira, W. Erlhagen, and E. Bicho, "Multi-bump solutions in a neural field model with external inputs," Physica D: Nonlinear Phenomena **326**, 32–51 (2016).
15. S. Mishra and R. Molinaro, "Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs II: A class of inverse problems," arXiv:2007.01138 **640**, 1–35 (2020).
16. W. Erlhagen and E. Bicho, "The dynamic neural field approach to cognitive robotics," Journal of neural engineering **3**, R36 (2006).
17. J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." Journal of machine learning research **13** (2012).