

An Efficient Architecture Selection Approach for PINNs Applied to 2D Electromagnetic Problems

Rodrigo Silva Rezende¹, Lara Celina Haase¹, Albert Piwonski¹, and Rolf Schuhmann¹

¹Theoretische Elektrotechnik, Technische Universität Berlin, Berlin, Germany

Physics-Informed Neural Networks (PINNs) can mitigate the difficulty that arises when using mesh-based approaches, such as finite difference methods, regarding the scalability of the model with bigger meshes. Since PINNs are mesh-free, they do not suffer from the same problems as mesh-based methods. However, choosing the PINNs' parameter is cumbersome. We propose a bottom-to-top method to compute the parameters of PINNs faster than random or surrogate-based optimization strategies.

Index Terms—Computational Modeling, Physics-informed Neural Networks, Partial Differential Equations, Radiofrequency

I. INTRODUCTION

Mesh-based methods, such as finite element method (FEM), are still the most used approaches to numerically solve partial differential equations (PDEs). They deliver low error and stable convergence rates, which justifies their establishment over the years as the best way to solve PDEs numerically.

Nevertheless, they have drawbacks, such as the solution's resolution, which can be numerically expensive, especially for high wavenumbers in propagation problems [1]. Also, their numerical costs for higher-dimensional problems generally scale badly. Further, they struggle while handling non-linearities.

Recently, neural networks (NN) have been used to solve PDEs. These models are called Physics-Informed Neural Networks (PINNs) [2]. The idea of PINNs is to add the PDE information and the BCs into the loss function. Hence, the PDE is converted into an optimization problem. PINNs' results are promising since they are mesh-free and can model high-dimensional geometrically complex non-linear problems [3].

However, choosing the right PINN for the problem at hand is complex since these models are defined by the # layers and neurons in each layer, its learning rate, the loss function, etc. Hence, the choice of these hyperparameters is itself a high-dimensional optimization process. If it is not done correctly the PINNs may not perform as desired.

A possible solution for this is to use a parameter optimization procedure. Many techniques can be used for this, such as random grid optimization or Gaussian processes (GPs)-based procedures [1]. Unfortunately, these are usually slow, since they may train and test overly complex architectures, which solve the problem, but require high computational costs. Often simpler architectures can also do the task. Yet the methods above can miss the simpler choices as a result of their approaches.

In this paper, we propose a bottom-to-top approach to fastly define a PINN and solve a specific PDE. This involves searching the model with the least # parameters possible, still capable of solving the problem. For this, we train and test less complex architectures and gradually increment the parameters of the PINNs until we reach the desired error.

As an application example, the inhomogeneous Helmholtz Equation for the vector potential $\mathbf{A}$ is solved with an imposed

current density $\mathbf{J}$ as excitation. We show that by using the bottom-to-top PINN method one can faster achieve more accurate results than the ones computed with a GP-based approach.

II. PINNs

Let, $\mathbf{x} \in \mathbb{R}^d$ be the input of the objective function we want to approximate. This is a function $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}^m := \mathbf{x} \mapsto \mathbf{y}$. For that, we use an NN with L layers and n_i neurons in the i -th layer, with $i = 1, 2, \dots, L$. This network can be simply described as $\mathbf{f}^{[L]} : \mathbb{R}^d \rightarrow \mathbb{R}^m := \mathbf{x} \mapsto \mathbf{y}^{[L]}$, with [3]:

$$\begin{cases} \mathbf{y}^{[0]} = \mathbf{x}, \\ \mathbf{y}^{[\ell]} = \sigma \circ (\mathbf{W}^{[\ell]} \mathbf{y}^{[\ell-1]} + \mathbf{b}^{[\ell]}), \end{cases} \text{ for } \ell = 1, 2, 3, \dots, L, \quad (1)$$

where $\mathbf{W}^{[\ell]} \in \mathbb{R}^{n_{i+1} \times n_i}$ and $\mathbf{b}^{[\ell]} \in \mathbb{R}^{n_{i+1}}$ are weights and biases of the ℓ -th layer. The σ represents the sigmoid function and $\circ$ is the pointwise product. Then, for a given L and n_i we want to compute the best $\mathbf{W}^{[\ell]}$ and $\mathbf{b}^{[\ell]}$ such that:

$$\arg \min_{\mathbf{W}^{[\ell]}, \mathbf{b}^{[\ell]}} \left(\mathbf{y}^{[L]} - \mathbf{y} \right)^2. \quad (2)$$

Now, consider we want to add some known physical law information in the cost function described in (2). This information is a PDE and involves our objective function $\mathbf{f}$, as follows:

$$\begin{aligned} \mathcal{N}[\mathbf{f}] &= \mathbf{u} \quad \text{in } D, \\ \mathcal{B}[\mathbf{f}] &= \mathbf{w} \quad \text{on } \partial D, \end{aligned} \quad (3)$$

where $\mathbf{u} : \mathbb{R}^d \rightarrow \mathbb{R}^m$ and $\mathbf{w} : \mathbb{R}^d \rightarrow \mathbb{R}^m$, D and ∂D are the computation domain and its boundaries, respectively. Note that we can build PDE and BC residuals with (1) and (3), as:

$$\begin{aligned} L_{PDE} &:= \mathcal{N}[\mathbf{f}^{[L]}] - \mathbf{u}, \\ L_{BC} &:= \mathcal{B}[\mathbf{f}^{[L]}] - \mathbf{w}, \end{aligned} \quad (4)$$

Finally, these residuals are incremented into (2):

$$\arg \min_{\mathbf{W}^{[\ell]}, \mathbf{b}^{[\ell]}} \left(\left(\mathbf{y}^{[L]} - \mathbf{y} \right)^2 + L_{PDE}^2 + L_{BC}^2 \right). \quad (5)$$

This final equation is a mathematical description of a PINN. More specifically, if we want to solve the inhomogeneous Helmholtz equation in the context of electromagnetism, the

input dimension is $d = 3$ and output $m = 3$. The function $\mathbf{f} := \mathbf{A}$, i.e., the vector potential. Then the PDE looks like:

$$\begin{aligned}\mathcal{N}[\mathbf{A}] &:= \Delta \mathbf{A} + \omega^2 \epsilon_0 \mu_0 \mathbf{A} = -\mu_0 \mathbf{J} \\ \mathcal{B}[\mathbf{A}] &:= \mathbf{A}|_{\partial D} = \mathbf{0},\end{aligned}\quad (6)$$

where $\mathbf{J}$ is the imposed current density. Note that, in (6) we define the BC as a Dirichlet BC equal to zero since it is the only BC we use here. Then our minimization problem becomes:

$$\arg \min_{\mathbf{w}^{[l]}, \mathbf{b}^{[l]}} \left(\left(\mathbf{A}^{[L]} - \mathbf{A} \right)^2 + \left(\Delta \mathbf{A}^{[L]} + \omega^2 \epsilon_0 \mu_0 \mathbf{A}^{[L]} + \mu_0 \mathbf{J} \right)^2 + \left(\mathbf{A}^{[L]}|_{\partial D} \right)^2 \right). \quad (7)$$

Note that we compute the weights and biases but not the # layers or neurons. As mentioned before, these values are crucial. Hence, to compute them, we suggest the following bottom-to-top algorithm: we initialize the # layers ($l = l_0$) and neurons ($n = n_0$). Here, we will only search for architectures that have the same # neurons on every layer. Then, we iterate over layers (l) and neurons (n). For each combination, we build a PINN model with the given parameters l and n , and compute the error. If error $< 7 \times 10^{-5}$, the process terminates. Otherwise, we increment n by 35 and repeat until the maximum # neurons is reached. Once the maximum # neurons is reached, we reset $n = n_0$, increment l by 1, and repeat the process until the maximum # layers is reached or the error criterion is met. Mind that the mentioned values for updating the parameters are hyperparameters of this update scheme and must also be estimated. For the present problem, the mentioned values seem to be a reasonable choice. Note that, since we search the # layers and neurons from bottom to top, we avoid the time-consuming big models but still solve the problem with the desired error. Even though we build every architecture from bottom to top, these models are simpler, and testing them is not computationally expensive. In the next section, we use this workflow to train a PINN to solve the Helmholtz equation for the vector potential problem.

III. APPLICATION EXAMPLE

We use the workflow above to solve the 2D inhomogeneous Helmholtz equation in (6). The domain is a rectangle (xy plane) with dimensions $[-0.5, 0.5] \times [-0.6, 0.6]$ m and a cable inside (point $[0, 0]$) with radius $r = 0.02$ m in the z-direction. On the cable, we impose a $\mathbf{J} = 1$ A/m². Hence, we have a z component for $\mathbf{A}$ and $\mathbf{J}$ in (7). Both objects are made of PEC. Then, the BC in (6) is suitable. We also compare our workflow with a GP-based approach, which builds a surrogate of the PINN's solution, using exponential basis functions [4], and apply it to compute the PINN's parameters [1]. All PINN's solutions are implemented in Python with the Package DeepXDE [5], coded as stated in (7). For both methods, we use max. # neurons = 150 and layers = 8. The model errors are calculated using the l_2 norm. For the train error goal, we set $< 1 \cdot 10^{-6}$ and test error $< 5 \cdot 10^{-4}$. Mind again that these are also hyperparameters. These values were also used in different cases and seem to be good choices for a general class

of problems. For the GP-based algorithm, we run 15 iterations, and for our method, we use exactly the values written before. The results are shown in Table I. Note that our method finds an architecture that is more complex than the one found by the GP method. However, our approach is faster since it first searches for the simpler models until it finds the last one. In contrast, GP-based searches require more complex methods that cost a lot of training time. In the end, we managed to be faster. Finally, we take the two optimal solutions and compute the magnetic field with $H_{PINN} := \frac{\nabla \times \mathbf{A}_{PINN}}{\mu_0}$. We then compute the reference value of the $\mathbf{H}$ field with an FD mesh-based in-house code and plot the H_x and H_y for both approaches against the FD reference. Note that the accuracy of both methods is almost the same, and we managed to do that in less time.

TABLE I
COMPARISON OF METHODS

Metric	Bottom-Top	GP-based
Training Time [s]	2489	3110
# Neurons (all layers)	60	38
# Layers	6	8
Test Error [l_2]	$2.21 \cdot 10^{-4}$	$3.87 \cdot 10^{-4}$

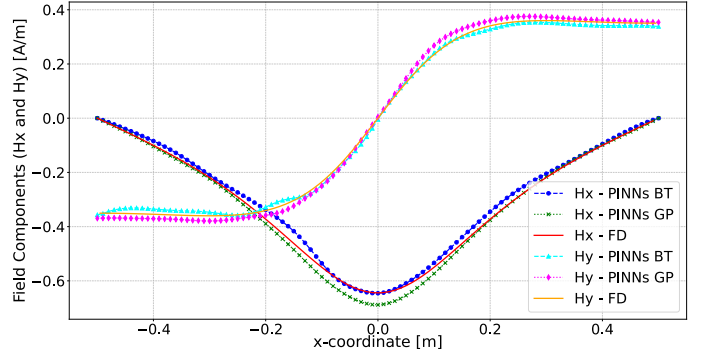


Fig. 1. H_x and H_y at $y=0.25$ for FD, bottom-top (BT), and GP

IV. CONCLUSION

We suggest a bottom-to-top approach to find the best parameters (layers and neurons) of a PINN's architecture. We benchmark our approach against a GP-based method. Using both solutions, we solve the 2D inhomogeneous Helmholtz equation with a source term. In the end, we managed to train the model faster and achieve similar results using our approach.

REFERENCES

- [1] P. Escapil-Inchauspé and G. A. Ruz, *Hyper-parameter tuning of physics-informed neural networks: Application to Helmholtz problems*, Neurocomputing, vol. 561, p. 126826, Dec. 2023.
- [2] 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 Comp. Physics, vol. 378, pp. 686–707, 2019.
- [3] X. Li, J. Deng, J. Wu, S. Zhang, W. Li, J. Deng, and Y.-G. Wang, *Physical informed neural networks with soft and hard boundary constraints for solving advection-diffusion equations using Fourier expansions*, arXiv:2306.12749v2, Dec. 2023.
- [4] R. S. Rezende, J. Hansen, A. Piwonski, and R. Schuhmann, *Wideband Kriging for Multiobjective Optimization of a High-Voltage EMI Filter*, IEEE Trans. on EMC. Comp., vol. 66, no. 4, pp. 1116–1124, Aug. 2024.
- [5] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, *DeepXDE: A deep learning library for solving differential equations*, SIAM Review, vol. 63, no. 1, pp. 208–228, 2021, <https://github.com/lululxvi/deepxde>.