SSN 1064-5624, Doklady Mathematics, 2024, Vol. 110, Suppl. 1, pp. S8–S14. © Pleiades Publishing, Ltd., 2024.
Russian Text © The Author(s), 2024, published in Doklady Rossiiskoi Akademii Nauk. Matematika, Informatika, Protsessy Upravleniya, 2024, Vol. 520, No. 2, pp. 11–18.

Search for Optimal Architecture of Physics-Informed Neural Networks Using Differential Evolution Algorithm

F. A. Buzaev[a,b,*], D. S. Efremenko[a,b], I. A. Chuprov[a,b], Ya. N. Khassan[a,b], E. N. Kazakov[b], and J. Gao[b]

Received August 8, 2024; revised October 2, 2024; accepted October 2, 2024

Abstract—The accuracy of solving partial differential equations using physics-informed neural networks (PINNs) significantly depends on their architecture and the choice of hyperparameters. However, manually searching for the optimal configuration can be difficult due to the high computational complexity. In this paper, we propose an approach for optimizing the PINN architecture using a differential evolution algorithm. We focus on optimizing over a small number of training epochs, which allows us to consider a wider range of configurations while reducing the computational cost. The number of epochs is chosen such that the accuracy of the model at the initial stage correlates with its accuracy after full training, which significantly speeds up the optimization process. To improve efficiency, we also apply a surrogate model based on a Gaussian process, which reduces the number of required PINN trainings. The paper presents the results of optimizing PINN architectures for solving various partial differential equations and offers recommendations for improving their performance.

Keywords: physics-informed neural networks, partial differential equations, genetic algorithms, differential evolution

DOI: 10.1134/S1064562424602300

1. INTRODUCTION

Physics-informed neural networks (PINNs) are a promising method for solving partial differential equations. These networks are trained using physical laws expressed as corresponding equations [1] and are often considered meshless alternatives to finite difference or finite element methods [2]. Today, the PINN method finds wide application in various fields such as computational fluid dynamics [3], nonlinear optics [4], heat transfer [5] and others, due to its versatility and the ability to be applied to virtually any equations with minimal changes in the code.

However, despite its widespread use, several unresolved issues remain. One of them is the dependence of the efficiency of a given PINN architecture on the specific equation for which it is applied. So far, no universal rule has been proposed for choosing the optimal PINN structure. For example, the AutoPINN method presented in [6] assumes sequential optimization of parameters based on empirical rules, but its optimality has not been proven and may vary depending on the problem.

Finding the optimal PINN configuration can be viewed as a global optimization problem, where the search space includes the hyperparameters of the PINN architecture and the function to be minimized is either the actual error or a loss function indirectly related to the actual error. To solve such problems, global optimization algorithms can be applied, which consider the optimized function as a black box. One of the effective methods of nonconvex optimization is the differential evolution algorithm [7], which has proven its effectiveness on a variety of benchmarks [8].

The aim of this work is to study the efficiency of using the differential evolution method for optimizing PINN and to develop practical recommendations for finding the best PINN architecture. To reduce the search time, a combined approach is used, including differential evolution and Gaussian process, which allows achieving accurate solutions in an acceptable time frame. Differential evolution searches the parameter space of a Gaussian process, which in turn selects the hyperparameters of a neural network. The proposed approach is tested on a set of equations including the Poisson, Helmholtz, and Burgers equations. The paper discusses the advantages of the proposed approach and its potential application in other areas of physics and engineering that require high accuracy and speed of modeling.

[a] *HSE University, Moscow, Russia*
[b] *Huawei Russian Research Center, Moscow, Russia*
**e-mail: fbuzaev@gmail.com*

2. ARCHITECTURES OF PHYSICS-INFORMED NEURAL NETWORKS

Physics-informed neural networks approximate solutions to partial differential equations (PDEs) using deep neural networks. The main characteristics of PINN can be described as follows. Consider a PDE in the following form:

$$\mathrm{F}(U(\mathbf{x})) = 0 \quad \text{for} \quad \mathbf{x} \in D, \tag{1}$$

where F is a differential operator, U is a solution to the PDE, and $\mathbf{x} = \{x_1, \ldots, x_n\} \in D$ is an n-dimensional vector of coordinates belonging to the region D. The operator F may include high-order partial derivatives of U with respect to x_i. The region D is bounded by Γ. The function U is subject to boundary conditions (here, we do not distinguish between boundary and initial conditions)

$$\mathrm{B}(U(\mathbf{x})) = 0 \quad \text{for} \quad \mathbf{x} \in \Gamma, \tag{2}$$

where B is a boundary operator. Our goal is to find the vector $\mathbf{b}$, which contains the parameters of the neural network such that PINN satisfies Eqs. (1) and (2). The desired vector is sought as a solution to the following optimization problem:

$$\mathbf{b} = \arg\min_{\mathbf{b}} L(\mathbf{b}), \tag{3}$$

where

$$L(\mathbf{b}) = L_{\mathrm{pde}}(\mathbf{b}) + L_{\mathrm{bc}}(\mathbf{b}), \tag{4}$$

$$L_{\mathrm{pde}}(\mathbf{b}) = \sum_{i=1}^{N} [\mathrm{F}(\mathrm{PINN}(\mathbf{x}_i, \mathbf{b}))]^2 \quad \text{for} \quad \mathbf{x}_i \in D, \tag{5}$$

$$L_{\mathrm{bc}}(\mathbf{b}) = \sum_{i=1}^{M} [\mathrm{B}(\mathrm{PINN}(\mathbf{x}_i, \mathbf{b}))]^2 \quad \text{for} \quad \mathbf{x}_i \in \Gamma, \tag{6}$$

where N and M are the numbers of sampling points inside D and on Γ, respectively, in which L_{pde} and L_{bc} are calculated. The partial derivatives of PINN with respect to elements x in Eq. (5) can be calculated using automatic differentiation tools implemented in machine learning libraries.

In early works, the PINN architecture was based on fully connected neural networks, which consist of multiple layers of neurons, each of which is connected to every neuron in the previous and next layers. This architecture provided sufficient flexibility to solve a wide range of problems, as it could easily adapt to different types of PDEs. In [9] it was shown that the accuracy of the solution to the equation, depending on the chosen architecture and its hyperparameters, can differ by several orders of magnitude. In this paper, we selected FNO, SPINN, and sinusoidal transform architectures for analysis, which are briefly described below.

One of the significant improvements in the PINN architecture was the Fourier neural operators (FNOs) [10]. FNO uses Fourier transforms to represent and process data, which allows for significantly faster computation. In this architecture, neural operators operate in Fourier space, where convolutions are performed in the frequency domain. This allows for efficient modeling of complex multivariate functions and improves the speed of model convergence. FNOs are particularly useful for problems that require processing data in infinite-dimensional function spaces, opening up new possibilities for solving complex PDEs. The use of FNOs in PINN is described in [11]. The corresponding model is called physics-informed neural operators (PINO).

Another important development in the PINN architecture is separable physics-informed neural networks (SPINN) [12]. SPINNs are based on the idea of variable separation. In SPINN, each solution is represented as a product of the outputs of several subnetworks, where each subnetwork is trained separately and is responsible for one of the problem variables. This results in a linear scaling of the problem complexity with the number of variables, in contrast to the exponential scaling in classical architectures. SPINNs demonstrate high efficiency in problems with separate variables and can significantly speed up the network training process.

Training an accurate PINN model often encounters difficulties due to local minima, which slows down the optimization process. In [13], it is proposed to use a sinusoidal mapping of the input data (sin) to increase the variability of the gradients, which helps to avoid such pitfalls. This method allows adapting to high-frequency patterns of the problem, thus improving training efficiency for a wide range of direct and inverse modeling problems in various physical domains.

Note that the above architectures can be used in combination with each other. For example, we consider SFNO (i.e., separable FNO) architectures with and without sinusoidal mapping.

3. HYBRID ALGORITHM OF DIFFERENTIAL EVOLUTION WITH A GAUSSIAN PROCESS

To find the optimal PINN configuration, this paper uses a hybrid algorithm combining differential evolution (DE) and Gaussian process (GP) [14]. The main idea of this approach is to use a Gaussian process as a surrogate model, which helps to significantly reduce the number of calls to the expensive objective function. In the initial stages of the algorithm, a Gaussian process is used to roughly estimate the values of the candidate fitness function. This allows reducing the number of expensive calculations of the real objective function. During the optimization process, the selection of candidates is carried out not only on the basis of the best predicted values of the fitness function, but also taking into account the uncertainty of these predictions. This approach avoids premature

convergence to local minima and supports more efficient exploration of the solution space.

The advantages of the hybrid DE-GP approach include significant reduction in computational cost, improvement in algorithm convergence due to uncertainty management, and the ability to scale the method to solve high-dimensional problems. This approach is particularly effective when solving problems with expensive fitness functions and complex search spaces [15–17].

Note that, in principle, it is possible to search for the optimal configuration without using a Gaussian process as a surrogate model. However, in this case we would be talking about a very labor-intensive task, namely, the need to train a number of configurations equal to the product of the population size of the DE algorithm (in our case 20) and the number of DE epochs (in our case 100).

The search for the optimal configuration is carried out as follows. First, the search space of neural network hyperparameters is initialized, including, if necessary, hyperparameters associated with the optimizer and training scheduler.

The Gaussian process is used as a surrogate model to approximate the relationship between hyperparameters and model training results. In this context, the hyperparameter search space acts as input features, and the objective function models the mean square error (if the reference solution is known) or the loss function (4).

To reduce computational costs at the initial stage, neural network training is carried out on 5% of the total number of epochs, which allows quickly assessing the potential of various combinations of hyperparameters. After completing the initial optimization using the surrogate model, the top five candidates are selected based on the predictions of the Gaussian process. These candidates are then fully trained over the entire number of epochs, resulting in final hyperparameter values that provide the best model performance.

Thus, the proposed approach combines the advantages of the Gaussian process for modeling complex dependences in the hyperparameter space and the efficiency of the differential evolution algorithm for finding optimal solutions, while providing a significant reduction in computational costs at the initial stages of optimization.

The hyperparameter search space used to optimize the PINN configuration includes the following parameters:

• Activation function between layers: Variations of activation functions applied between layers of a neural network include sine, tanh, ReLU, GeLU, and tanhshrink.

• Number of layers: The number of hidden layers in the network varies from 1 to 8.

• Layer width (number of neurons in each layer): Different hidden layer width configurations are considered: {50, 100, 200, 300, 400, 500}.

• Using a free parameter as additional information: The possibility of including or excluding a free parameter as an additional input to the model is considered: {True, False}.

The hyperparameter search space used to optimize the FNO configuration includes the following parameters:

• Activation function between layers: The activation functions sine, tanh, ReLU, GeLU, and tanhshrink are considered.

• Number of layers: The numbers of layers considered are: {1, 2, 3}.

• Using a free parameter as additional information: Possible values: {True, False}.

• Layer width: Layer width options: {4, 8, 16, 32, 64}.

• Number of mods: The mode numbers considered are: {2, 4, 8, 16, 32}.

The hyperparameter search space used to optimize the SPINN configuration includes the following parameters:

• Activation function between layers: The activation functions sine, tanh, ReLU, GeLU, and tanhshrink are considered.

• Number of layers: Number of layers considered: {1, 2, 3, 4, 5, 6, 7, 8}.

• Layer width: Layer width options: {50, 100, 200, 300, 400, 500}.

• Using a free parameter as additional information: Possible values: {True, False}.

• Mix the free parameter with coordinate embeddings: Possible values: {True, False}.

The hyperparameter search space used to optimize the SFNO configuration includes the following parameters:

• Activation function between layers: The activation functions sine, tanh, ReLU, GeLU, and tanhshrink are considered.

• Number of layers: Possible numbers of layers: {1, 2}.

• Layer width: Layer width options: {4, 8, 16, 32, 64}.

• Number of mods: Possible mod numbers: {2, 4, 8, 16, 32}.

• Using a free parameter as additional information: Possible values: {True, False}.

• Mix the free parameter with coordinate embeddings: Possible values: {True, False}.

If we also include the optimizer and training scheduler hyperparameters in the optimization process, the

Table 1. MSE values of the Poisson equation solution obtained during the optimization of PINNs configurations

Model configuration	Loss function	MSE
PINN	4.4e–5 (9e–5)	1.9e–5 (1.3e–4)
PINN+sin	3.1e–2 (2.2e–2)	1.2e–3 (6.5e–3)
FNO	1.7e–4 (2.3e–4)	6.2e–5 (5.1e–4)
FNO+sin	9.1e–5 (1.8e–4)	2.6e–5 (1.6e–4)
SPINN	1.7e–4 (2.3e–4)	6.3e–5 (1.4e–2)
SPINN+sin	5e–6 (8.4e–5)	3.2e–6 (5.1e–4)
SFNO	2e–6 (1e–6)	3e–6 (1.1e–5)
SFNO+sin	3.1e–6 (1.2e–4)	7.8e–7 (1.0e–5)

The values in parentheses correspond to the case where the search space includes the hyperparameters of the optimizer and scheduler of the training.

following additional space is added to the main search space:

• Learning rate: The learning speed is selected from the set: {1e-5, 1e-4, 1e-3, 1e-2, 1e-1}.

• Weight decay: The regularization parameter L2 (weight penalty) is selected from the set: {1e-5, 1e-4, 1e-3, 1e-2, 1e-1}.

• Learning rate reduction factor: The factor by which the learning rate is reduced when a plateau is detected, with possible values: {1e-3, 1e-2, 1e-1, 5e-1, 9e-1}.

• Threshold for reducing learning rate: The threshold values at which learning rate reduction is activated include: {1e-3, 1e-2, 1e-1, 5e-1, 9e-1}.

• Patience (delay): The number of epochs without improvement after which the learning rate decreases, with possible values: {5, 10, 20, 50, 100, 200}.

4. NUMERICAL EXPERIMENTS

4.1. Training Configuration

All computational experiments were performed on an Nvidia V100 GPU with 16 GB of memory. It should be noted that one epoch of training neural networks with Fourier neural operators requires an order of magnitude more time. To ensure comparability of results, training was limited to the number of epochs: 20 000 for PINN and SPINN models and 2000 for FNO and SFNO models. The AdamW algorithm was used as the optimization method, without additional optimization of the optimizer and scheduler hyperparameters, with an initial learning rate of 0.001. To adapt the learning rate, the standard ReduceLROnPlateau scheduler was used, which decreased it by 10% if there was no improvement in the loss function for 50 consecutive epochs. The network weights were initialized using Xavier's method to uniformly distribute the initial parameters. Each experiment used 2500 sample points, randomly and uniformly distributed across the domain under study, which remained fixed throughout the training process to ensure consistency of results.

The search for the optimal configuration of models was carried out on the basis of minimizing the loss function or mean square error (MSE). In case of using MSE, a reference solution is required, which implies the need to precalibrate the model based on known solutions before applying it to new problems. If there is no reference solution, minimization of the loss function is possible. However, the loss function serves only as an indirect criterion for the accuracy of the PINN method, and its minimization does not always guarantee the most accurate solution.

4.2. The Poisson Equation

The Poisson equation is an elliptic partial differential equation and describes the potential created by an electrostatic charge. The equation has the following form:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \tag{7}$$

where u is an unknown function and f is the source function. In our work f is the following polynomial:

$$f(x, y) = (1 - x^2)(2y^3 - 3y^2 + 1). \tag{8}$$

Figure 1 shows the dependence of MSE on the corresponding values obtained after 5% of the total number of training epochs. Some correlation can be seen between the metrics at 5 and 100% epochs. Moreover, architecture selection using MSE minimization is comparable in efficiency to architecture selection using loss function minimization. Also in the legend, red color means the normal space and blue color means the sinusoidal space. The name of the elements in the legend contains the plus sign, which means that the search space also includes the search space of the optimizer and scheduler hyperparameters.

The obtained MSE values for each configuration are given in Table 1. Note that including the optimizer and planner hyperparameters in the optimization space leads to improved results only for the fully connected PINN.

4.3. The Helmholtz Equation

Let us apply the method described in the article to the Helmholtz equation given by

$$\nabla^2 u = -k^2 f. \tag{9}$$

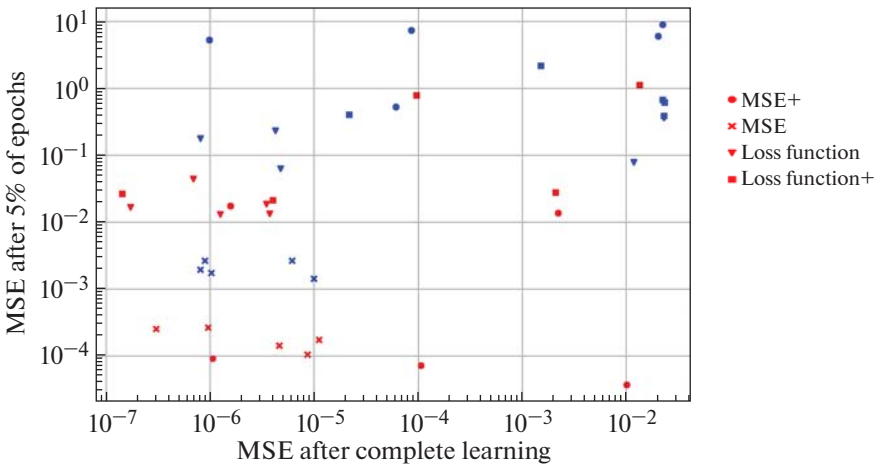

Fig. 1. Mean square error (MSE) over the corresponding values obtained at 5% of the training epochs for the Poisson equation. Results are shown for the top 5 models for each training scenario. Red dots correspond to training in the sinusoidal space, blue dots are for training without sinusoidal space. The plus sign means that the search space also includes the search space of the optimizer and scheduler hyperparameters.

For our tests we consider the two-dimensional case:

$$-u_{xx} - u_{yy} - k_0^2 u = f(x,y), \tag{10}$$

where k and f are the wavenumber and the source of the elliptical PDE, respectively. The solution is sought for the source function in the form

$$f(x,y) = \sin(k_0 x)\sin(k_0 y), \tag{11}$$

in the domain $\Omega = [0,1]^2$ with the Dirichlet boundary conditions $u(x,y) = 0$, $(x,y) \in \partial\Omega$. The value k_0 is 4π.

The learning results are shown in Fig. 2. The results of the best experiments for all model configurations and variations of the optimized metric are shown in Table. 2. The best result is highlighted in bold. Using a sinusoidal transform does not provide any obvious advantage.

4.4. The Burgers Equation

The Burgers equation arises in various areas of applied mathematics, including fluid mechanics, nonlinear acoustics, and gas dynamics. It can be obtained from the Navier–Stokes equation by eliminating the pressure gradient term. For our tests we consider the following equation:

$$u_t + uu_x = \left(\frac{0.01}{\pi}\right)u_{xx}, \quad \text{in the domain} \quad x \in [0,1]^2, \quad t \in [0,1]^2 \tag{12}$$

with the initial conditions $u(0,x) = -\sin(\pi x)$ and the Dirichlet boundary conditions $u(t,-1) = u(t,1) = 0$.

The training results are shown in Fig. 3 and Table 3. The SFNO architecture turned out to be the most accurate, with the other architectures showing accuracy 4–5 orders of magnitude lower.

Table 2. MSE values of the Helmholtz equation solution obtained during the optimization of PINNs configurations

Model configuration	Loss function	MSE
PINN	8.1e–1 (9.6e–1)	8.7e–1 (7.9e–1)
PINN+sin	7.4e–1 (7.9e–1)	1.2e–2 (8.5e–2)
FNO	7.2e–2 (7.9e–2)	1.8e–2 (6.3e–2)
FNO+sin	8.3e–2 (7.9e–2)	2.3e–2 (5.9e–1)
SPINN	7.8e–2 (8.2e–1)	7.8e–2 (7.9e–2)
SPINN+sin	7.7e–2 (7.8e–2)	6.6e–2 (8.3e–2)
SFNO	8.6e–5 (4.3e–5)	6.5e–5 (9.5e–5)
SFNO+sin	7.2e–5 (4.9e–4)	2e–6 (6.5e–4)

The values in parentheses correspond to the case where the search space includes the hyperparameters of the optimizer and the training scheduler.

Table 3. MSE values of the Burgers equation solution obtained during the optimization of PINNs configurations

Model configuration	Loss function	MSE
PINN	3.6e–2 (2.9e–2)	5.6e–2 (7.3e–2)
PINN+sin	2.9e–2 (2.9e–2)	4.5e–2 (4.9e–2)
FNO	9.6e–3 (1.5e–3)	1.3e–3 (1.2e–3)
FNO+sin	9.2e–3 (1.9e–3)	1.1e–3 (1.4e–3)
SPINN	1.5e–2 (1.4e–2)	1.4e–2 (1.4e–2)
SPINN+sin	1.2e–2 (1.6e–2)	8.9e–3 (1.3e–2)
SFNO	1.7e–2 (2.4e–2)	3.0e–2 (1.2e–2)
SFNO+sin	2.7e–2 (1.2e–2)	7.8e–2 (9.8e–2)

The values in parentheses correspond to the case where the search space includes the hyperparameters of the optimizer and the training scheduler.

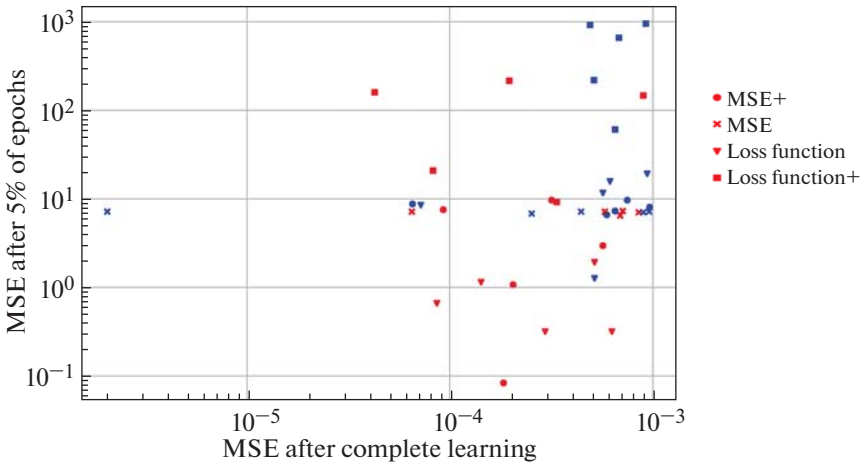

Fig. 2. The same as in Fig. 1, but for the Helmholtz equation.

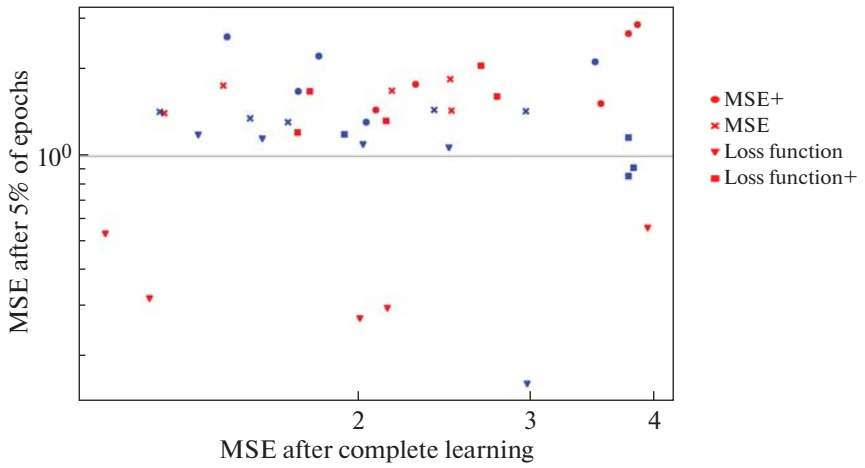

Fig. 3. The same as in Fig. 1, but for the Burgers equation.

5. CONCLUSIONS

In this paper, optimization methods of physics-informed neural networks (PINNs) using a hybrid algorithm of differential evolution (DE) and Gaussian process (GP) were investigated in detail. The main objective of the study was to find the optimal configuration of PINN model architecture for solving partial differential equations (PDEs) such as the Poisson, Helmholtz, and Burgers equations.

The use of the hybrid DE-GP approach allowed us to significantly reduce the number of expensive calculations of the objective function, which significantly increased the efficiency of the optimization process. An important feature was the optimization carried out on a small number of PINN training epochs, which did not reduce the accuracy of the model, but significantly accelerated the search process.

The results showed that none of the PINN architectures used is universally the best; their accuracy is highly dependent on the chosen hyperparameters. To achieve the highest accuracy, it makes sense to optimize all the architectures under consideration and then select the best one. It was also demonstrated that, although the loss function is an indirect criterion for evaluating the accuracy of the PINN, its minimization can become the basis for finding the optimal architecture and corresponding hyperparameters.

As for the optimization of the hyperparameters of the optimizer and scheduler itself, no clear conclusion could be made. On the one hand, their adjustment can increase the accuracy of the model, and, on the other hand, it increases the dimensionality of the problem and complicates the optimization process.

Future research is planned to expand to include more complex neural network architectures and different types of PDEs. Also promising is the application of the developed method to solve real engineering problems where high accuracy of modeling and optimization is required.

FUNDING

This work was supported by ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. S. Kollmannsberger, D. D'Angella, M. Jokeit, and L. Herrmann, "Physics-informed neural networks," *Deep Learning in Computational Mechanics* (Springer International, Cham, 2021), pp. 55–84.
2. S. Markidis, "The old and the new: Can physics-informed deep-learning replace traditional linear solvers?" Front. Big Data **4**, 669097 (2021).
3. P. Wu, K. Pan, L. Ji, S. Gong, W. Feng, W. Yuan, and C. Pain, "Navier–Stokes generative adversarial network: A physics-informed deep learning model for fluid flow generation," Neural Comput. Appl. **34** (14), 11539–11552 (2022).
4. I. Chuprov, Jiexing Gao, D. Efremenko, E. Kazakov, F. Buzaev, and V. Zemlyakov, "Optimization of physics-informed neural networks for solving the nonlinear Schrödinger equation," Dokl. Math. **108** (Suppl. 2), S186–S195 (2023).
5. S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," J. Heat Transfer **143** (6), (2021).
6. Y. Wang, X. Han, C.-Y. Chang, D. Zha, U. Braga-Neto, and X. Hu, "Auto-PINN: Understanding and optimizing physics-informed neural architecture" (2022).
7. R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," J. Global Optim. **11** (4), 341–359 (1997).
8. M. F. Ahmad, N. A. M. Isa, W. H. Lim, and K. M. Ang, "Differential evolution: A recent review based on state-of-the-art works," Alexandria Eng. J. **61** (5), 3831–3872 (2022).
9. F. Buzaev, J. Gao, I. Chuprov, and E. Kazakov, "Hybrid acceleration techniques for the physics-informed neural networks: A comparative analysis," Mach. Learn. **113** (6), 3675–3692 (2023).
10. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations" (2020).
11. Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, "Physics-informed neural operator for learning partial differential equations (2021).
12. J. Cho, S. Nam, H. Yang, S.-B. Yun, Y. Hong, and E. Park, "Separable physics-informed neural networks" (2023).
13. J. C. Wong, C. C. Ooi, A. Gupta, and Y.-S. Ong, "Learning in sinusoidal spaces with physics-informed neural networks," IEEE Trans. Artif. Intell. **5** (3), 985–1000 (2024).
14. D. Petelin, B. Filipic, and J. Kocijan, *Optimization of Gaussian Process Models with Evolutionary Algorithms* (Springer, Berlin, 2011), pp. 420–429.
15. H. T. Rauf, W. H. K. Bangyal, and M. I. Lali, "An adaptive hybrid differential evolution algorithm for continuous optimization and classification problems," Neural Comput. Appl. **33** (17), 10841–10867 (2021).
16. H. Yu, L. Kang, Y. Tan, J. Zeng, and C. Sun, "A multimodel assisted differential evolution algorithm for computationally expensive optimization problems," Complex Intell. Syst. **7** (5), 2347–2371 (2021).
17. X. Chen and A. Shen, "Self-adaptive differential evolution with Gaussian–Cauchy mutation for large-scale CHP economic dispatch problem," Neural Comput. Appl. **34** (14), 11769–11787 (2022).

Publisher's Note. Pleiades Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
AI tools may have been used in the translation or editing of this article.