EVOLUTIONARY TWO-STAGE HYPERPARAMETER OPTIMIZATION STRATEGIES FOR PHYSICS-INFORMED NEURAL NETWORKS

Fedor Buzaev[1], **Dmitry Efremenko**[1], **Egor Bugaev**[1], **Andrei Ermakov**[1,2], **Denis Derkach**[1], **Daria Pugacheva**[*1,2], **Fedor Ratnikov**[*1]
[1]HSE University, [2]AXXX

ABSTRACT

Physics-Informed Neural Networks (PINNs) solve Partial Differential Equations (PDEs) by embedding physical laws into neural network training. However, their performance suffers from unstable convergence, training plateaus, and strong sensitivity to architectural and optimization hyperparameters due to the highly non-convex and multi-term structure of the physics-informed loss. In this setting, the outer-loop hyperparameter search is a noisy and black-box optimization problem over heterogeneous parameters, where classical local or gradient-based strategies are easily trapped in suboptimal regions. Evolutionary algorithms, with their population-based exploration and ability to handle mixed, non-differentiable search spaces, provide a more robust mechanism for discovering promising configurations. We propose and investigate a two-stage approach based on evolutionary algorithms that combines exploration and exploitation parts of PINNs training to improve solution accuracy and robustness under fixed computational budgets. In the first stage, we perform low-fidelity training runs with truncated epochs to rapidly screen candidate configurations, treating hyperparameter selection as a black-box outer-loop problem. In the second stage, only the most promising candidates are fully trained with standard gradient-based optimizers to refine the solution. Evaluated on three popular problems, namely Advection, Klein–Gordon and Helmholtz equations, our method consistently outperforms standard training and achieves significantly lower mean error within constrained computational resources.

1 INTRODUCTION

Physics-Informed Neural Networks (PINNs) solve PDEs by minimizing a loss that includes residuals of the governing equations (Raissi et al., 2019). They have been applied in fluid dynamics, heat transfer, and quantum mechanics. Despite this promise, PINNs training often stalls, diverges, or converges to suboptimal minima, with robustness poorly understood (Krishnapriyan et al., 2021; Wang et al., 2022a; Chuprov et al., 2024; Buzaev et al., 2023; 2024). To compete with classical solvers like finite differences Mao et al. (2020); Grossmann et al. (2023), PINNs must be made more efficient and stable.

A central obstacle is sensitivity to hyperparameters. Architectural choices and optimization settings can drastically alter convergence behavior, causing training plateaus or convergence to poor local minima. This sensitivity is particularly challenging because hyperparameter tuning is naturally bilevel: for each hyperparameter configuration, an inner gradient-based optimizer trains the network weights, while an outer procedure must select the hyperparameters that yield the best final solution quality. In practice, this outer-loop objective is a computationally expensive black box, since model quality is only observed after executing a training run. Moreover, the loss landscape of PINNs has been compared to that of standard deep neural networks (DNNs), revealing a significantly higher density of local minima Wong et al. (2025); Wang et al. (2022a). This increased multimodality is cited as a primary reason why PINN training typically requires more iterations to converge and is

[*]Equal advising.

prone to suboptimal solutions. Consequently, classical local strategies may prematurely commit to suboptimal regions of the search space.

These considerations motivate the use of population-based evolutionary optimization in the outer loop. Evolutionary algorithms can search effectively in heterogeneous hyperparameter spaces Peng et al. (2021); Bäck et al. (2023), where discrete architectural decisions interact with continuous optimization parameters (e.g., learning rates, scheduler parameters, and loss-term weights), without requiring differentiability of the outer objective. Their population-based nature also supports a practical exploration–exploitation trade-off. Rather than fully training many configurations to convergence, one can rapidly filter a large set of candidates and then concentrate computation on the most promising ones. This is especially important under fixed computational budgets. Most prior works rely on ad hoc tuning for individual problems (Buzaev et al., 2023). Lacking systematic strategies, they may waste substantial resources on configurations that reach plateaus early.

We propose a two-phase approach that integrates global optimization methods into PINN training. First, we evaluate the quality of solutions for different hyperparameter sets over a small number of epochs and select the most promising configurations for further training in the second phase. We assessed the performance of different evolutionary algorithms for selection, such as JADE Zhang & Sanderson (2009), LSHADE Mohamed et al. (2017), Grey Wolf Mirjalili et al. (2014), and Whales Mirjalili & Lewis (2016). A comparison against classical approaches, including Random Search, Grid Search, the Nelder–Mead Nelder & Mead (1965) method, and Bayesian optimization, on a popular and practically significant set of equations, namely Advection, Klein–Gordon and Helmholtz equations, shows that the proposed method enables systematic discovery of improved solutions.

Thus, the main contribution is as follows:

1. The proposed two-stage optimization strategy enables reliable identification of high-performing PINN configurations under strict computational budgets.
2. We show that population-based evolutionary algorithms consistently outperform classical hyperparameter tuning methods such as Bayesian optimization, random search, and grid search on benchmark PDEs.
3. We formulate evidence-based guidelines for optimal budget distribution between exploration and exploitation phases, demonstrating that an exploration budget of approximately 10% of standard training achieves about 40% average improvement of the baseline-level error value under the fixed computational budget.

Our results provide practical guidelines for robust PINN training, reducing reliance on manual tuning and enhancing applicability to complex physical systems.

2 Related work

Several studies have addressed the hyperparameter sensitivity of Physics-Informed Neural Networks (PINNs) through systematic or automated tuning approaches. Grid search has been applied to explore discrete hyperparameter combinations Ren et al. (2022), while Bayesian optimization has gained popularity for its sample efficiency in high-dimensional spaces, particularly for problems such as the Helmholtz equation Escapil-Inchauspé & Ruz (2022). However, as the number of hyperparameters increases (including architecture, activation functions, learning rates, and loss weights), these methods become computationally prohibitive due to the expensive black-box evaluations required for each configuration.

Manual tuning across a broad range of hyperparameters has also been explored Cho et al. (2022). Such hand-crafted configurations often transfer reasonably well to similar problems within the same physical domain, serving as effective starting points but lacking generalizability and requiring significant expert effort.

Neural Architecture Search (NAS) techniques have been adapted specifically to PINNs. For instance, Auto-PINN Wang et al. (2022b); Ahmad et al. (2026) employs NAS to automatically optimize network architecture and other hyperparameters, achieving notable accuracy improvements. Similarly, NAS-PINN and related frameworks automate the search for optimal PINN designs Wang

& Zhong (2023). While effective, these approaches remain computationally intensive, as they require numerous full training runs to evaluate candidate architectures.

To mitigate the high cost of full-fidelity evaluations, low-fidelity approximations—such as training networks for fewer epochs—have been proposed in the broader DNN literature Elsken et al. (2018) and references therein. These surrogate-based strategies enable faster screening of configurations during hyperparameter optimization. However, their application to PINNs, particularly in combination with NAS or evolutionary methods, remains underexplored, and practical guidelines for balancing fidelity, exploration, and final accuracy are largely absent.

Recent surveys highlight the growing use of evolutionary algorithms for PINN hyperparameter and architecture optimization Wong et al. (2025), motivated by their robustness in noisy, non-convex, and mixed search spaces. Nevertheless, most existing works either perform full training for every candidate or lack a structured separation between rapid global exploration and high-fidelity exploitation.

To our knowledge, this is among the first works to conduct a head-to-head comparison of multiple state-of-the-art evolutionary optimizers (including JADE, LSHADE, Grey Wolf Optimizer, and Whale Optimization Algorithm) within a practical two-stage PINN tuning pipeline. Our approach differs from prior methods (e.g. Zhang & Yang (2024); Wong et al. (2023); Penwarden et al. (2022); Howard et al. (2023); Zhou et al. (2025)) in that it explicitly combines population-based global search on low-fidelity (truncated-epoch) training runs for efficient exploration with full-gradient-based refinement of only the most promising candidates in the exploitation phase. This structured allocation of computational budget enables systematic discovery of superior configurations while achieving significant time savings compared to full-training baselines or NAS-only strategies.

3 BACKGROUND

3.1 PINN METHODOLOGY

PINNs solve PDEs by combining the approximation capabilities of neural networks with the physical constraints defined by PDEs and their boundary conditions. Consider a bounded domain $\Omega \subset \mathbb{R}^d$ with boundary $\partial\Omega$, governed by a PDE system:

$$\begin{aligned} \mathcal{D}u(\mathbf{x}) &= s(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ \mathcal{B}u(\mathbf{x}) &= g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{1}$$

where $u : \mathbb{R}^d \to \mathbb{R}^m$ is the solution field, $\mathcal{D}$ a differential operator, $\mathcal{B}$ is the boundary operator, $s(\mathbf{x})$ and $g(\mathbf{x})$ are known functions. To approximate the true solution $u(\mathbf{x})$, we introduce a neural network $\hat{u}(\mathbf{x}; \psi)$, parameterized by trainable weights and biases ψ. This network serves as a surrogate for $u(\mathbf{x})$, and its parameters ψ are optimized to satisfy both the PDE and the boundary conditions. In particular, the training process involves minimizing a composite loss function that quantifies how well the neural network adheres to the physical constraints. The total loss is defined as:

$$L(\psi) = L_{\text{PDE}}(\psi) + L_{\text{BC}}(\psi), \tag{2}$$

where $L_{\text{PDE}}(\psi)$ is the PDE residual loss, which enforces the PDE within the domain Ω:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} \|\mathcal{D}\hat{u}(\mathbf{x}_i) - s(\mathbf{x}_i)\|^2 \tag{3}$$

and $L_{\text{BC}}(\psi)$ is the boundary condition loss, which enforces the boundary conditions on $\partial\Omega$:

$$\mathcal{L}_{\text{bc}} = \frac{1}{N_{\partial\Omega}} \sum_{j=1}^{N_{\partial\Omega}} \|\mathcal{B}\hat{u}(\mathbf{x}_j) - g(\mathbf{x}_j)\|^2 \tag{4}$$

In these expressions, N_Ω and $N_{\partial\Omega}$ represent numbers of collocation points sampled from the interior of the domain, and from the boundary, respectively. The optimal parameters ψ of PINN are found by minimizing the total loss during PINN training:

$$\psi^* = \arg\min_{\psi} L(\psi). \tag{5}$$

This optimization is typically carried out using gradient-based algorithms, such as Adam or L-BFGS, which use gradients of $L(\psi)$ with respect to ψ, computed via automatic differentiation. By minimizing the loss, the neural network learns to approximate the PDE solution while adhering to the boundary conditions, effectively recasting the PDE problem as an optimization task.

3.2 Optimization with Evolutionary Algorithms

We study several global evolutionary optimization algorithms, each based on different principles and strategies. Below is a brief description of each algorithm. Building on the framework presented in Buzaev et al. (2024), this study extends the methodological toolbox to six separate evolutionary algorithms, allowing for a comprehensive evaluation of optimization performance.

The Grey Wolf Optimizer (GWO) is a nature-inspired metaheuristic that emulates the leadership hierarchy and hunting mechanism of grey wolves in nature. The algorithm categorizes wolves into four groups, representing the social hierarchy. The hunting process involves three main steps: encircling prey, hunting, and attacking prey (Mirjalili et al., 2014).

The Whale Optimization Algorithm (WOA) is inspired by the bubble-net hunting strategy of humpback whales. It simulates the whales' behavior of creating bubble nets to encircle prey. The algorithm includes encircling prey, bubble-net attacking method (exploitation phase), and search for prey (exploration phase) (Mirjalili & Lewis, 2016).

JADE is an adaptive variant of the Differential Evolution (DE) algorithm that incorporates self-adaptive control parameters and an optional external archive to enhance performance. It adjusts the mutation factor and crossover rate dynamically based on historical information, improving convergence speed and solution quality (Zhang & Sanderson, 2009; Buzaev et al., 2024).

LSHADE is an advanced version of the Success-History Adaptive Differential Evolution (SHADE) algorithm, integrating Lévy-flight-based mechanisms to enhance exploration capabilities. It employs a historical memory of successful parameter settings and adapts the population size during the optimization process. These features enable LSHADE to perform robustly on complex, high-dimensional optimization problems (Tanabe & Fukunaga, 2014).

4 Two-stage strategy of PINN hyperparameter optimization

The PINN approximation $\hat{u}(\mathbf{x}; \boldsymbol{\psi}, \boldsymbol{H})$ introduces two parameter classes:

- $\boldsymbol{\psi} \in \mathbb{R}^p$: Trainable network weights/biases
- $h \in \mathcal{H}$: Hyperparameters (learning rates, loss weights $\lambda_{\text{pde}}, \lambda_{\text{bc}}$, network architecture)

The training process constitutes two coupled optimization tasks:

$$\begin{aligned} &\text{(Inner)} \quad \min_{\boldsymbol{\psi}} \mathcal{L}(\boldsymbol{\psi}; h) = \lambda_{\text{pde}} \mathcal{L}_{\text{pde}} + \lambda_{\text{bc}} \mathcal{L}_{\text{bc}}, \\ &\text{(Outer)} \quad \min_{h \in \mathcal{H}} \|\hat{u}(\mathbf{x}; \boldsymbol{\psi}^*(h), h) - u_{\text{ref}}(\mathbf{x})\|^2, \end{aligned} \tag{6}$$

where $\boldsymbol{\psi}^*(\boldsymbol{H})$ denotes weights optimized with fixed $\boldsymbol{H}$, and $u_{\text{ref}}(\mathbf{x})$ is a reference solution used for calibrating the hyperparameters. If a reference solution is not available, the outer optimization task may be taken as follows:

$$\text{(Outer)} \quad \min_{\boldsymbol{H}} \mathcal{L}(\boldsymbol{\psi}; \boldsymbol{H}). \tag{7}$$

The proposed two-phase optimization framework builds upon foundational advances in automated neural network design and efficient hyperparameter tuning Elsken et al. (2019).

The computational budget is allocated across two phases. The first phase involves an outer optimization process, referred to as the exploration phase, where multiple PINNs are trained with different hyperparameter configurations to identify the most promising one. At this stage, full training of each PINN is not necessarily required; instead, it is sufficient to estimate the potential of each configuration to yield the best performance. Once the most promising configuration is identified, the exploitation phase begins, during which the selected PINN is fully trained to completion.

The search space for PINN hyperparameters encompasses both network architecture and optimization settings. For architecture, variations consisted of changing activation functions, the number layers and units per layer. For optimization, we explored different scheduling schemes and learning rate ranges.

Only the parameters relevant to the chosen scheduler type are activated during the search, which reduces the effective dimensionality of the space and speeds up the black-box optimizer. By activating only the parameters relevant to the selected scheduler, the optimizer explores a lower-dimensional search space, and thus converges more efficiently.

The coupled optimization problem defined in Equation equation 6 constitutes a bilevel optimization task. While classical bilevel optimization assumes convexity and smoothness for theoretical guarantees, PINN training inherently involves non-convex loss landscapes due to the neural network's nonlinear parameterization. This renders exact gradient-based hyperparameter optimization intractable, motivating our two-phase approach that reformulates the outer loop as a black-box optimization problem.

Our framework addresses this by separating exploration and exploitation:

- **Exploration phase:** Global optimizers efficiently prune suboptimal regions of $\mathcal{H}$ using limited training epochs, using population-based search to avoid local minima.
- **Exploitation phase:** Full training of the best configuration refines the solution, capitalizing on the neural network's local convergence properties.

5 Numerical results

To assess the performance of our optimization-assisted PINN framework, we consider three benchmark partial differential equations: the flow mixing problem, the Klein–Gordon equation, and the Helmholtz equation. We provide additional details for these equations in Appendix A. These problems are well-established in scientific computing and are known to present significant challenges for PINN-based solvers Krishnapriyan et al. (2021). For rigorous validation, we adopt the strategy introduced in (Cho et al., 2023), wherein an exact solution is assumed and substituted into the governing equation. The resulting residual is treated as a source term, yielding a modified problem for which the exact solution is known by construction. This approach enables direct and quantitative evaluation of the PINN's accuracy against a ground-truth reference.

5.1 Experimental setup

We used for training 1000 collocation points, 100 points for boundary condition and 100 points for initial condition. We set the following range of hyperparameters:

- **Architecture**
  - number of layers: $2-5$
  - units per layer: $16-128$
  - activation function $\in\{$`tanh`, `ReLU`, `GELU`, `SiLU`$\}$
- **Optimiser (AdamW)**
  - learning rate: $10^{-4}-10^{-2}$ (log-uniform)
- **Learning-rate scheduler**
  - `scheduler_type` $\in\{$`reduce_on_plateau`, `cosine_annealing`, `none`$\}$
  - *If* `scheduler_type = reduce_on_plateau`:
    - `scheduler_factor` $\in[0.1,\ 0.5]$
    - `scheduler_patience` $\in\{5,\ 20\}$

All computational experiments were performed on a single NVIDIA Tesla V100 GPU with 32 GB of memory. Each optimization iteration (10,000 training epochs) required approximately 11 minutes on average. For the exploration phase, we used 200 iterations per optimization run and 5 independent random seeds per equation. A population size of 30 is used across all optimizers.

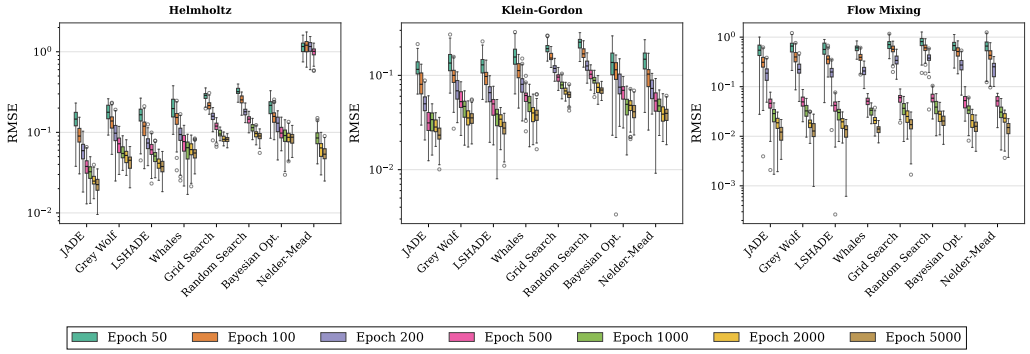

Figure 1: RMSE distribution across hyperparameter optimization methods for the Helmholtz (top), Klein-Gordon (middle), and flow mixing (bottom) equations at different epoch budgets (50–5000 epochs per trial, color-coded) with fixed total num of the iterations. Error decreases linearly (log scale) starting from 500 epochs, indicating optimal exploration budget.

5.2 Preliminary analysis

In this section, we investigate the amount of training epochs are sufficient to reliably identify the most efficient configuration at the end of training.

There is an inherent trade-off when choosing a reduced cutoff for training epochs: optimization algorithms must discriminate which configuration is ultimately the best, but waiting until full convergence is computationally expensive. Conversely, if training is prematurely truncated, the configuration that appears best early on may not remain the best by the end.

To study this behavior, we generated multiple configurations through random search and six global optimization algorithms. The resulting evaluation curves for three problems, namely Helmholtz3D, Klein-Gordon3D, and FlowMixing2D.

From the analysis, it is evident that to reliably identify the absolute best configuration, almost all out of 10000 epochs are typically needed (please see Appendix B for the details). However, requiring such a long training phase significantly increases the computational burden. In practice, the selection criteria may be relaxed: instead of demanding the absolute best, it is acceptable to select a configuration that finishes better than 90% of all trials in terms of final error.

Under this relaxed criterion, it is sufficient to use approximately 200 epochs to assess and select a potentially high-performing PINN configuration. Early selection strategies with modest cutoff epochs (e.g., 200) can dramatically accelerate the optimization workflow without severely compromising final accuracy. In this way, the method of lower fidelity estimates (Elsken et al., 2019) and learning curve extrapolation (Swersky et al., 2014) are extended to PINN.

We address the task of finding optimal architecture for the PINN model to solve a particular problem.

The number of trials in the exploration phase thus is driven by the nature of the original problem, and often drives the computing resources to solve the problem. Quality of the undertrained model for the given set of hyperparameters may be a good proxy to estimate the ultimate model quality for this set of parameters. Our hypothesis is that for PINN driven problems the rate of convergence of the model performance as a function of number of training epochs is similar for different sets of model hyperparameters. We tested this hypothesis for three PINN benchmark problems. Figure 1 demonstrates that indeed after a moderate number of training epochs we get a good approximation for the final model quality. We found that 500 initial training epochs is a good training period to conclude, if the model with this set of hyperparameters could get to 10% of top models.

Another important question concerns the number of iterations to perform during the exploration phase. We investigate the error reduction for varying numbers of hyperparameter optimization iterations during full retraining. As shown in Figure 2, after screening approximately 150 configurations

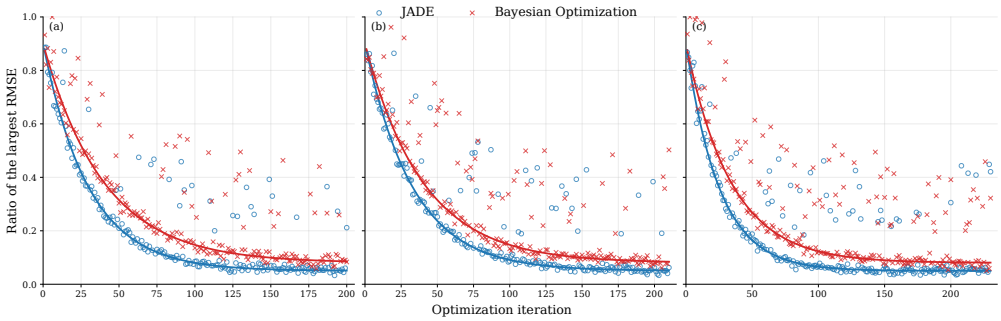

Figure 2: Saturation of the optimization process for JADE and Bayesian Optimization across three benchmark PDEs: Helmholtz (top), Klein–Gordon (middle), and flow mixing (bottom). The vertical axis shows the ratio of the current best RMSE to the final best RMSE. Dashed vertical line indicates iteration 150, where saturation is observed across all problems.

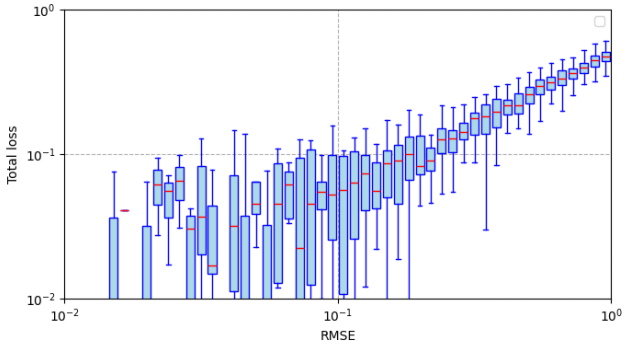

Figure 3: Relationship between the root mean square error (RMSE) and the total loss for the Helmholtz equation

additional iterations yield diminishing returns. The computational time of 32 GPU hours spent on such optimization will serve as our baseline in conditions of limited computational budget.

5.3 Two-Stage Optimization

We apply global optimization solvers to obtain accurate PINN solutions. In each global optimization algorithm, we assume a fixed evaluation budget, defined as the total number of trials allowed during the exploration phase. Each trial corresponds to training a PINN from 100 to 10000 epochs, where 10000 is the classical optimization with full learning. As an objective (cost) function, either the value of the total loss function or the root mean square error (RMSE) – if a reference solution is available – can be used. Importantly, RMSE correlates with the total loss (as demonstrated in 4 and Figure 3), which in practice allows to predict the final error based on the loss value. Therefore, we distinguish between two optimization regimes. When the reference solution is available, the optimization targets direct calibration of the PINN by minimizing the RMSE. When the reference solution is not available, the optimization proceeds by minimizing the total loss function.

Figure 4 shows a comparison of the efficiency of different global optimization algorithms across three problems. Two optimization strategies are considered: (1) direct minimization of the RMSE; (2) minimization of the PINN loss function.

Although optimization based on the loss function generally leads to slightly worse results compared to RMSE minimization, it remains more practical in cases where a reference solution is not available. For all considered equations, JADE performs best, whereas all examined evolutionary algorithms demonstrate performance that is either competitive with or superior to both Bayesian optimization and Nelder–Mead.

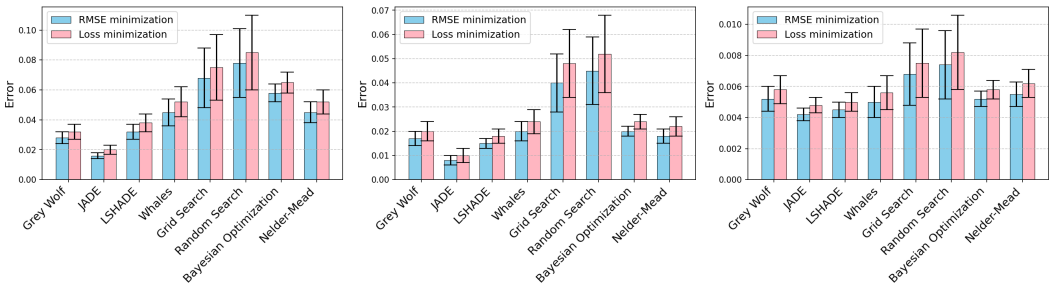

Figure 4: Global hyperparameter search with population-based evolutionary optimizers (Grey Wolf, JADE, LSHADE, Whales) significantly outperforms traditional baseline methods (Grid Search, Random Search, Bayesian Optimization, Nelder-Mead) across all three PDEs after 10,000 training epochs. On the Helmholtz equation (left), Klein-Gordon equation (middle), and flow mixing equation (right), evolutionary algorithms consistently achieve lower prediction errors. Results averaged over five runs; error bars show standard deviation.

Figure 5 presents distributions of the minimal RMSE for the final trained configurations selected after conducting the exploration phase with the fixed baseline computational budget and varying number of trials and epochs (100, 200, 500, 1000, 2000, 5000, and 10000) across several optimization strategies for the Klein-Gordon equation. As can be seen from the Figure, for all evolutionary based optimizers, the minimum error is achieved with the number of exploratory epochs of 10^3, which is less than full convergence. Evolutionary algorithms, in particular JADE, reach the most optimal values and outperform the Bayesian approach.

Figure 6 shows the computational efficiency of two-phase hyperparameter optimization for three benchmark equations. Baseline corresponds the error value of one-stage JADE optimization for full 10,000 epochs of retraining. As can be seen from the Figure, JADE optimization achieves the minimum at 1,000 epochs with error reduction of 77% (Helmholtz), 73% (Klein–Gordon), and 28% (Flow Mixing) in comparison with one-stage baseline.

JADE consistently outperforms the Bayesian optimization across all tasks. The performance gap between methods widens with increasing epochs and at 5,000 epochs JADE achieves 0.26 compared to 1.17 for Bayesian Optimization on the Helmholtz problem.

Thus, the evolutionary algorithm JADE more effectively explores the multi-modal hyperparameter landscape of PINNs due to its population-based search and adaptive adjustment of mutation parameters, whereas Bayesian Optimization becomes trapped in local optima of the surrogate model.

The best configurations and results demonstrating that the influence of individual hyperparameters is weak are shown in Appendix C.

6 Conclusions

In this work, we propose a two-stage framework that integrates global optimization strategies into the training of Physics-Informed Neural Networks (PINNs). The first stage, exploration, uses evolutionary algorithms, such as JADE, LSHADE, Grey Wolf and WOA, as global optimization solvers to efficiently search for promising hyperparameter configurations with a reduced number of training epochs. In the second stage, exploitation, the selected configuration is trained fully to convergence.

We evaluate the framework on three benchmark PDE problems, namely the Helmholtz equation, the Klein–Gordon equation, and the Flow Mixing problem. In all cases, optimization-assisted PINNs demonstrate superior accuracy compared to baseline one-stage training strategies. Solutions found through global optimization achieve significantly lower RMSEs and are more robust to local minima, as evidenced by improved convergence curves and more accurate reconstruction of nonmonotonic solution features. We demonstrate that RMSE correlates strongly with the total PINN loss (Figure 3 and 4), enabling practical hyperparameter selection even when reference solutions are unavailable. The best configurations identified architectural preferences (for instance, GELU acti-

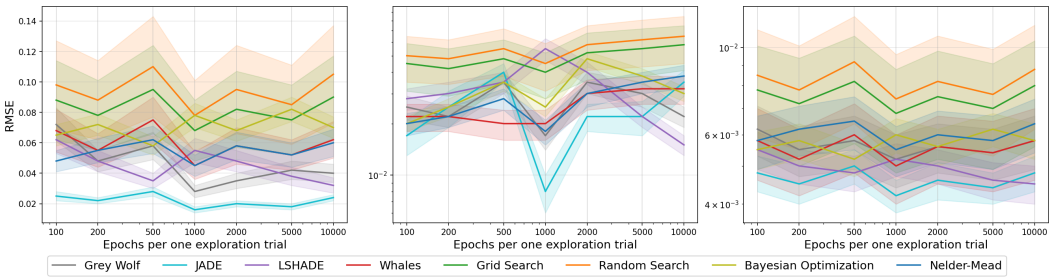

Figure 5: Effect of exploration epoch budget on PINN accuracy for a fixed total hyperparameter search time across the Helmholtz equation (left), Klein-Gordon equation (middle, log scale), and flow mixing equation (right, log scale). The X-axis shows the number of epochs allocated per exploration trial (100–10,000), while the Y-axis displays the final RMSE achieved after retraining. Each curve represents one of eight global optimization algorithms. For each epoch budget, the optimizer explored the hyperparameter space using truncated training runs, then the 10 best configurations were retrained to convergence. Shaded regions indicate standard deviation across five independent runs.

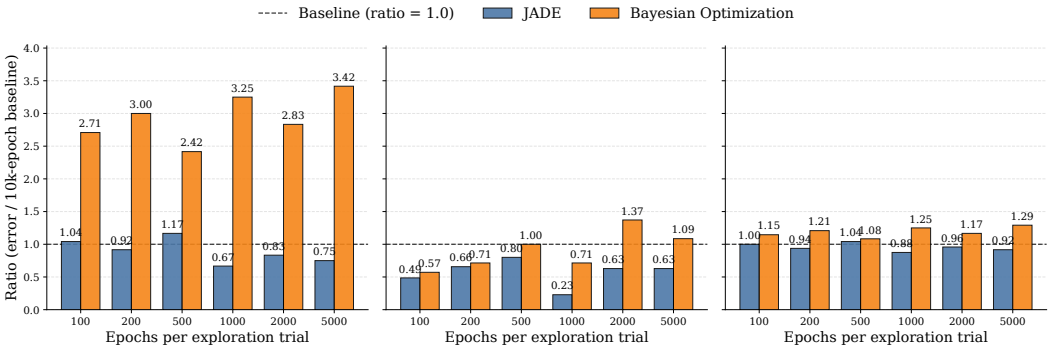

Figure 6: Computational efficiency of two-phase hyperparameter optimization for the Helmholtz equation (top), Klein-Gordon equation (middle), and flow mixing equation (bottom). The x-axis shows epochs per exploration trial; the y-axis shows the RMSE ratio to baseline (10,000 epochs, dashed line at 1.0). Values below 1.0 indicate better accuracy for the same computational budget. JADE (blue) consistently outperforms Bayesian optimization (pink), achieving 40% average improvement at 1,000 exploration epochs, with up to 77% error reduction for Klein-Gordon. The advantage diminishes at higher epoch budgets as both methods approach baseline performance.

vation for Helmholtz and flow mixing, SiLU for Klein–Gordon), though individual hyperparameter effects remain intertwined and resist simple interpretation.

Consistent with the No Free Lunch theorem Wolpert & Macready (1997), our experiments reveal no single global optimization solver that emerged as a universal solution across all considered PDE classes. However, differential evolution-based methods (with the best result for JADE) demonstrate superior stability in final solution accuracy compared to classical Bayesian approaches, random and grid search (Figure 4,5 and 6). This empirical advantage likely stems from their inherent adaptation mechanisms (such as parameter self-tuning) being particularly well-suited to the structured non-convexity of PINN loss landscapes.

Our experiments demonstrate that an exploration phase consisting of 10% of the epochs required for full training per trial is sufficient to reliably select high-performing PINN configurations, improving model quality by 28% to 77% for the considered problems compared to the classical scheme with full fine-tuning per trial. Short exploration phases (about 1-2% of the total epochs) are less reliable and require more fine-tuning, diminishing overall efficiency. Exceeding 20% of epochs leads to diminishing returns. Thus, 10% training epochs provide the best balance between efficiency

and model quality, establishing structured hyperparameter optimization as an effective approach for resource-efficient and robust PINNs.

Future work will focus on extending the optimization framework to a broader search space that includes variations in PINN architectures and broader PDE families.

ACKNOWLEDGMENTS

The work was supported by the grant for research centers in the field of AI provided by the Ministry of Economic Development of the Russian Federation in accordance with the agreement 000000C313925P4E0002 and the agreement with HSE University 139-15-2025-009.
This research is supported in part through computational resources of HPC facilities at HSE University.

REFERENCES

Ahmad, Husna Zafar, Aneeqa Zafar, Muhammad Noveel Sadiq, A.K. Awasthi, Homan Emadifar, and Karim K. Ahmed. Evolution of physics-informed neural networks: Recent architectural variants and optimization strategies. *Array*, 29:100688, March 2026. ISSN 2590-0056. doi: 10.1016/j.array.2026.100688. URL http://dx.doi.org/10.1016/j.array.2026.100688.

F. A. Buzaev, D. S. Efremenko, I. A. Chuprov, Ya. N. Khassan, E. N. Kazakov, and J. Gao. Search for optimal architecture of physics-informed neural networks using differential evolution algorithm. *Doklady Mathematics*, 110(S1):S8–S14, December 2024. ISSN 1531-8362. doi: 10.1134/s1064562424602300. URL http://dx.doi.org/10.1134/S1064562424602300.

Fedor Buzaev, Jiexing Gao, Ivan Chuprov, and Evgeniy Kazakov. Hybrid acceleration techniques for the physics-informed neural networks: a comparative analysis. *Machine Learning*, 113(6): 3675–3692, December 2023. ISSN 1573-0565. doi: 10.1007/s10994-023-06442-6. URL http://dx.doi.org/10.1007/s10994-023-06442-6.

Thomas H. W. Bäck, Anna V. Kononova, Bas van Stein, Hao Wang, Kirill A. Antonov, Roman T. Kalkreuth, Jacob de Nobel, Diederick Vermetten, Roy de Winter, and Furong Ye. Evolutionary algorithms for parameter optimization—thirty years later. *Evolutionary Computation*, 31(2):81–122, 06 2023. ISSN 1063-6560. doi: 10.1162/evco_a_00325. URL https://doi.org/10.1162/evco_a_00325.

Gyouho Cho, Di Zhu, Jeffrey Joseph Campbell, and Mengqi Wang. An lstm-pinn hybrid method to estimate lithium-ion battery pack temperature. *IEEE Access*, 10:100594–100604, 2022. ISSN 2169-3536. doi: 10.1109/access.2022.3208103. URL http://dx.doi.org/10.1109/ACCESS.2022.3208103.

Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable physics-informed neural networks. *Advances in Neural Information Processing Systems*, 2023.

I. A. Chuprov, J. Gao, D. S. Efremenko, F. A. Buzaev, and V. V. Zemlyakov. Solution of the multimode nonlinear schrödinger equation using physics-informed neural networks. *Doklady Mathematics*, 110(S1):S15–S24, December 2024. ISSN 1531-8362. doi: 10.1134/s1064562424602105. URL http://dx.doi.org/10.1134/S1064562424602105.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. 2018. doi: 10.48550/ARXIV.1808.05377. URL https://arxiv.org/abs/1808.05377.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: a survey. *J. Mach. Learn. Res.*, 20(1):1997–2017, January 2019. ISSN 1532-4435.

Paul Escapil-Inchauspé and Gonzalo A. Ruz. Hyper-parameter tuning of physics-informed neural networks: Application to helmholtz problems, 2022. URL https://arxiv.org/abs/2205.06704.

Tamara G. Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method?, 2023. URL https://arxiv.org/abs/2302.04107.

Amanda A. Howard, Mauro Perego, George Em Karniadakis, and Panos Stinis. Multifidelity deep operator networks for data-driven and physics-informed problems. *Journal of Computational Physics*, 493:112462, November 2023. ISSN 0021-9991. doi: 10.1016/j.jcp.2023.112462. URL http://dx.doi.org/10.1016/j.jcp.2023.112462.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf.

Zhiping Mao, Ameya D. Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, March 2020. ISSN 0045-7825. doi: 10.1016/j.cma.2019.112789. URL http://dx.doi.org/10.1016/j.cma.2019.112789.

Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Advances in Engineering Software*, 95:51–67, May 2016. ISSN 0965-9978. doi: 10.1016/j.advengsoft.2016.01.008. URL http://dx.doi.org/10.1016/j.advengsoft.2016.01.008.

Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61, March 2014. ISSN 0965-9978. doi: 10.1016/j.advengsoft.2013.12.007. URL http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.

Ali W. Mohamed, Anas A. Hadi, Anas M. Fattouh, and Kamal M. Jambi. Lshade with semi-parameter adaptation hybrid with cma-es for solving cec 2017 benchmark problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 145–152, 2017. doi: 10.1109/CEC.2017.7969307.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, January 1965. ISSN 1460-2067. doi: 10.1093/comjnl/7.4.308. URL http://dx.doi.org/10.1093/comjnl/7.4.308.

Hu Peng, Yupeng Han, Changshou Deng, Jing Wang, and Zhijian Wu. Multi-strategy co-evolutionary differential evolution for mixed-variable optimization. *Knowledge-Based Systems*, 229:107366, 2021. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2021.107366. URL https://www.sciencedirect.com/science/article/pii/S0950705121006286.

Michael Penwarden, Shandian Zhe, Akil Narayan, and Robert M. Kirby. Multifidelity modeling for physics-informed neural networks (pinns). *Journal of Computational Physics*, 451:110844, February 2022. ISSN 0021-9991. doi: 10.1016/j.jcp.2021.110844. URL http://dx.doi.org/10.1016/j.jcp.2021.110844.

Maziar Raissi, Paris Perdikaris, and George 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.

Pu Ren, Chengping Rao, Yang Liu, Jian-Xun Wang, and Hao Sun. Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes. *Computer Methods in Applied Mechanics and Engineering*, 389:114399, February 2022. ISSN 0045-7825. doi: 10.1016/j.cma.2021.114399. URL http://dx.doi.org/10.1016/j.cma.2021.114399.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-Thaw Bayesian optimization, 2014. URL https://arxiv.org/abs/1406.3896.

Ryoji Tanabe and Alex S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1658–1665. IEEE, July 2014. doi: 10.1109/cec.2014.6900380. URL http://dx.doi.org/10.1109/CEC.2014.6900380.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, January 2022a. ISSN 0021-9991. doi: 10.1016/j.jcp.2021.110768. URL http://dx.doi.org/10.1016/j.jcp.2021.110768.

Yicheng Wang, Xiaotian Han, Chia-Yuan Chang, Daochen Zha, Ulisses Braga-Neto, and Xia Hu. Auto-pinn: Understanding and optimizing physics-informed neural architecture, 2022b. URL https://arxiv.org/abs/2205.13748.

Yifan Wang and Linlin Zhong. Nas-pinn: Neural architecture search-guided physics-informed neural network for solving pdes, 2023. URL https://arxiv.org/abs/2305.10127.

D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997. ISSN 1089-778X. doi: 10.1109/4235.585893. URL http://dx.doi.org/10.1109/4235.585893.

Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, Pao-Hsiung Chiu, Joshua Shao Zheng Low, My Ha Dao, and Yew-Soon Ong. Evolutionary optimization of physics-informed neural networks: Advancing generalizability by the baldwin effect, 2023. URL https://arxiv.org/abs/2312.03243.

Jian Cheng Wong, Abhishek Gupta, Chin Chun Ooi, Pao-Hsiung Chiu, Jiao Liu, and Yew-Soon Ong. Evolutionary optimization of physics-informed neural networks: Evo-pinn frontiers and opportunities, 2025. URL https://arxiv.org/abs/2501.06572.

Bo Zhang and Chao Yang. Discovering physics-informed neural networks model for solving partial differential equations through evolutionary computation. *Swarm and Evolutionary Computation*, 88:101589, July 2024. ISSN 2210-6502. doi: 10.1016/j.swevo.2024.101589. URL http://dx.doi.org/10.1016/j.swevo.2024.101589.

Jingqiao Zhang and A.C. Sanderson. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, October 2009. ISSN 1089-778X. doi: 10.1109/tevc.2009.2014613. URL http://dx.doi.org/10.1109/TEVC.2009.2014613.

Wanyun Zhou, Haoze Song, and Xiaowen Chu. Automated design for physics-informed modeling with convolutional neural networks. *Communications Physics*, 8(1), November 2025. ISSN 2399-3650. doi: 10.1038/s42005-025-02414-5. URL http://dx.doi.org/10.1038/s42005-025-02414-5.

A Considered equations

A.1 Flow mixing problem (Advection Equation)

We consider the two-dimensional linear advection equation:

$$\frac{\partial u(t,x,y)}{\partial t} + \frac{\partial u(t,x,y)}{\partial x} + \frac{\partial u(t,x,y)}{\partial y} = 0, \tag{8}$$

which describes the passive transport of a scalar field in a uniform velocity field with components $a = 1$, $b = 1$ along the x- and y-axes, respectively.

An exact solution to this equation is:

$$u(t,x,y) = \sin\left(\pi(x-t)\right)\sin\left(\pi(y-t)\right). \tag{9}$$

The computational domain is defined as $(x,y) \in [0,1]^2$, $t \in [0,1]$. The initial condition is given by:

$$u(0,x,y) = \sin(\pi x)\sin(\pi y), \tag{10}$$

which matches the exact solution at $t = 0$.

To ensure periodicity and compatibility with the exact solution, we impose periodic boundary conditions:

$$\begin{aligned} u(t, 0, y) &= u(t, 1, y), \\ u(t, x, 0) &= u(t, x, 1), \end{aligned} \tag{11}$$

for all $t \in [0, 1]$. These conditions maintain consistency of the transported field across the boundaries throughout the simulation.

A.2 Klein–Gordon Equation

The Klein–Gordon equation is a nonlinear hyperbolic PDE describing relativistic scalar field dynamics. In two spatial dimensions, it takes the form:

$$\begin{aligned} \frac{\partial^2 u(t,x,y)}{\partial t^2} - \frac{\partial^2 u(t,x,y)}{\partial x^2} - \frac{\partial^2 u(t,x,y)}{\partial y^2} + u(t,x,y)^2 \\ = f(t,x,y), \end{aligned} \tag{12}$$

where $(x, y) \in [-1, 1]^2$, $t \in [0, 10]$, and $f(t, x, y)$ is a source term defined by:

$$f(t,x,y) = u(t,x,y)^2 - 4\,u(t,x,y). \tag{13}$$

The exact solution is given by:

$$u_{\text{exact}}(t,x,y) = (x+y)\cos(2t) + xy\sin(2t). \tag{14}$$

The initial conditions are derived from the exact solution:

$$u(0,x,y) = u_{\text{exact}}(0,x,y), \quad \frac{\partial u}{\partial t}(0,x,y) = \frac{\partial u_{\text{exact}}}{\partial t}(0,x,y). \tag{15}$$

Similarly, Dirichlet boundary conditions are imposed using the exact solution:

$$u(t,\pm 1,y) = u_{\text{exact}}(t,\pm 1,y), \quad u(t,x,\pm 1) = u_{\text{exact}}(t,x,\pm 1). \tag{16}$$

A.3 Helmholtz equation

We consider the three-dimensional Helmholtz equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + k^2 u(x,y,z) = f(x,y,z), \tag{17}$$

where $u(x, y, z)$ is the scalar field, k is the wave number, $f(x, y, z)$ is a source term defined as

$$f(x,y,z) = \left[-3\pi^2 + k^2\right]\sin(\pi x)\sin(\pi y)\sin(\pi z). \tag{18}$$

Zero Dirichlet boundary conditions are imposed. Consequently, the analytic solution is given by

$$u_{\text{exact}}(x,y,z) = \sin(\pi x)\sin(\pi y)\sin(\pi z). \tag{19}$$

B Additional details for preliminary analysis

The results of convergence for three equations are presented in Figures 7.

In each figure, we plot two metrics:

- The average probability best-so-far remains best (right axis, orange curve), which quantifies the likelihood that a trial currently identified as best remains the best at the final epoch.

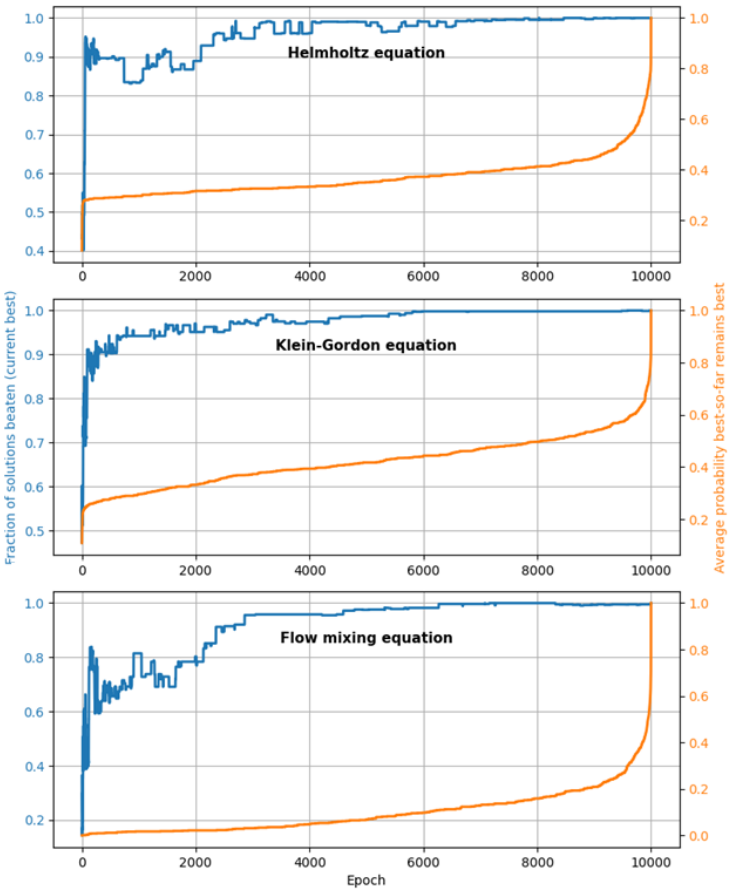

Figure 7: Prediction stability for the Helmholtz equation (top) , Klein-Gordon equation (middle) and flow mixing problem (bottom). Fraction of solutions beaten by the current best configuration and average probability best-so-far remains best as function of the current epoch. Results are averaged over 3000 PINN configurations.

- The fraction of solutions beaten (left axis, blue curve), which measures what fraction of all configurations the current best solution outperforms at the end of training.

All training processes were performed up to 10,000 epochs. The current leading configuration may change from epoch to epoch, resulting in visible jumps in the probability curves.

C CONFIGURATIONS

Table 1: Best found PINN configurations per problem for a 1000-epoch exploration phase with fixed computation time corresponding to 1279 parameter trials.

Parameter	**Helmholtz**	**Klein–Gordon**	**Flow Mixing**
Layers	4	5	3
Layer Size	128	94	109
Activation	GELU	SiLU	GELU
Learning Rate	9.1×10^{-3}	4.8×10^{-3}	8.1×10^{-3}
Scheduler	Reduce Plateau	Cosine Annealing	Reduce Plateau
Sch. Factor	0.5	110 ($T_{\max}$)	0.5
Sch. Patience	20	8.1×10^{-5} ($\eta_{\min}$)	16

The best configurations are presented in Table 1 and demonstrates the specificity of the obtained configurations for the different equations, where the optimum is achieved with different architectures and sets of hyperparameters.

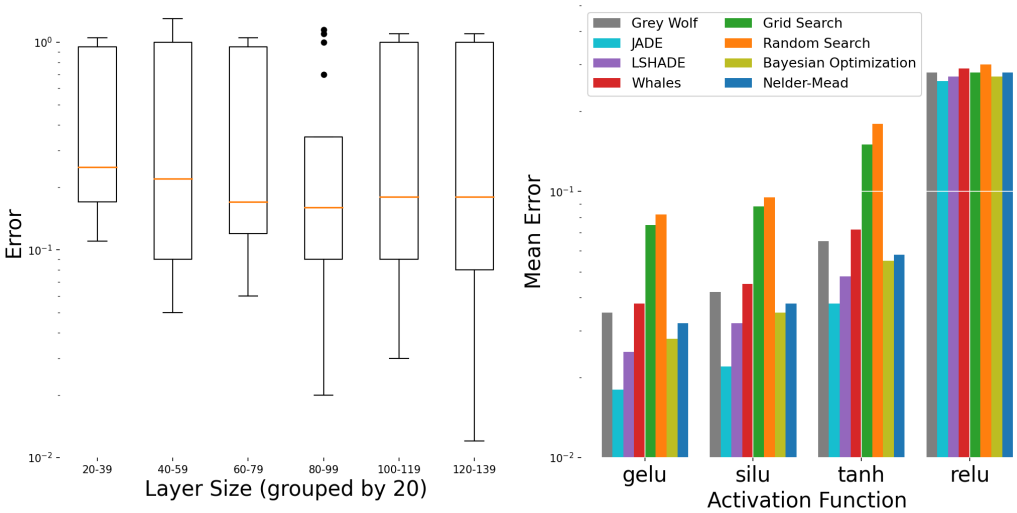

Figure 8: Mean error as a function of layer size and activation function, averaged across all considered trials and remaining hyperparameter settings for Helmholtz equation.

The influence of individual hyperparameters is on average quite weak. The results are significantly blurred by random initialization of weights and variations in other settings. However, the activation function (see Fig. 8) remains a relatively smooth and almost independent factor influencing its interpretation. At the opposite poles, the parameters of such a layer size have complex, interdependent effects that are closely intertwined with other hyperparameters. Such interrelationships complicate manual tuning and call into question the percentage of surrogate methods (e.g., Bayesian optimization), which have difficulty accurately capturing multivariate nonlinear dependencies. This, in turn, emphasizes the need for more flexible approaches.