PINNFORGE: EXECUTION-GROUNDED EVOLUTIONARY DESIGN OF PHYSICS-INFORMED NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-informed neural networks (PINNs) require coordinated choices across representation, architecture, sampling, constraints, loss construction, and optimization, yet effective configurations can vary substantially across partial differential equations (PDEs). Existing automated design methods search over these choices, but training outcomes are still used primarily to rank candidates rather than to inform subsequent designs. We develop PINNForge, an execution-grounded large language model (LLM)-driven evolutionary framework that treats observed PINN training behavior as a cross-generation design signal. PINNForge initializes diverse configurations from PDE-related prior knowledge, executes them, retains globally strong configurations, and feeds their execution evidence and accumulated run-level experience back to the LLM. The LLM then revises and recombines coupled design components or explores new combinations within a generate–execute–evaluate–evolve loop. Across 25 PDE benchmarks, PINNForge achieves the lowest mean MSE on 24 tasks among RoPINN, PINNsFormer, PINNsAgent, and PINNForge. Removing knowledge guidance, execution feedback, or evolutionary search increases the median task-wise MSE ratio to 3.74×, 12.10×, and 10.10×, respectively, relative to full PINNForge. Anonymous code: `https://anonymous.4open.science/r/PINNForge/`.

1 INTRODUCTION

Physics-informed neural networks (PINNs) solve partial differential equations (PDEs) by embedding governing equations and physical constraints into neural-network training (Raissi et al., 2019; Toscano et al., 2025; Bai et al., 2025; Michaloglou et al., 2026). In practice, however, their accuracy depends on a set of strongly coupled design choices, including input representation, network architecture, collocation sampling, constraint handling, loss construction, and optimization (Krishnapriyan et al., 2021; Rathore et al., 2024; Wang et al., 2021a; 2022; Wu et al., 2023; Wang et al., 2024). The dominant bottleneck can also differ substantially across PDEs: multiscale problems may demand more expressive representations, complex geometries may rely more heavily on sampling and constraint enforcement, and long-time systems may be limited by causality or optimization stability. Consequently, a configuration that performs well on one PDE may not transfer effectively to another.

This motivates automated PINN design. Existing work has explored neural architecture search, hyperparameter optimization, evolutionary computation, and LLM-based agents to reduce manual trial and error (Wang et al., 2023; Wang & Zhong, 2024; Kaplarević-Mališić et al., 2023; Zhang & Yang, 2024; Carrillo et al., 2024; Wuwu et al., 2025; You et al., 2026). These approaches demonstrate that important PINN design choices can be automated, but training outcomes are still used primarily to rank candidates or diagnose isolated failures. In contrast, execution provides richer evidence—including PDE residuals, physical-constraint errors, loss and gradient behavior, stagnation, and numerical stability—that can reveal not only which candidate performs better, but also why a configuration underperforms and which aspects of the design may require revision.

The challenge is further compounded by interactions among PINN components. A representation bottleneck, for example, may require coordinated changes to architecture, sampling, loss balancing, and optimization rather than an isolated numerical adjustment. Automated design therefore calls for a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

search mechanism that reasons over complete configurations, preserves effective mechanisms, revises coupled components in response to observed failures, and explores beyond the current elite set.

Recent LLM-driven algorithm-discovery methods provide a natural foundation for this direction. FunSearch, Evolution of Heuristics (EoH), ReEvo, and LLaMEA embed LLMs within iterative generate–execute–evaluate–evolve loops, using strong candidates and execution outcomes to guide subsequent proposals (Romera-Paredes et al., 2024; Liu et al., 2024; Ye et al., 2024; van Stein & Bäck, 2025; Yao et al., 2025; Liu et al., 2026). In contrast, evolutionary PINN design has largely relied on predefined representations together with conventional mutation and crossover operators (Kaplarević-Mališić et al., 2023; Zhang & Yang, 2024; Carrillo et al., 2024). This suggests a natural opportunity to combine LLM-based semantic reasoning with evidence from real PDE solving to evolve complete PINN configurations across generations.

We therefore propose PINNForge, an execution-grounded evolutionary framework for automated PINN design. It starts from diverse configurations informed by PDE characteristics and relevant PINN knowledge, then executes them through real training to obtain performance, physical, and optimization evidence. PINNForge feeds globally strong configurations, their execution evidence, and accumulated run-level experience back to the LLM, which uses five complementary strategies—Refine, Physics-Guided, Architecture-Guided, Synthesis, and Novelty—to inherit, revise, recombine, or explore coupled PINN components. In this way, candidate training becomes part of the subsequent design process rather than a terminal evaluation. The main contributions are as follows:

1. We formulate automated PINN design as an execution-grounded multi-generation search over a unified design space spanning representation, architecture, sampling, constraints, loss construction, optimization, and training control.
2. We develop an EoH-inspired LLM-driven evolutionary mechanism in which PDE-related prior knowledge, globally retained high-performing configurations, and execution evidence guide cross-component inheritance, revision, recombination, and exploration.
3. Across 25 heterogeneous PDEs, PINNForge achieves the lowest mean MSE on 24 tasks compared with RoPINN, PINNsFormer, and PINNsAgent. Controlled ablations, search trajectories, and training-budget analysis further examine the roles of its main search mechanisms.

2 Related Work

2.1 PINN Optimization and Automated Design

PINN training can be hindered by gradient imbalance, mismatched convergence rates across loss terms, insufficient sampling, spectral bias, inaccurate constraint enforcement, and ill-conditioned optimization (Wang et al., 2021a; 2022; Wu et al., 2023; Krishnapriyan et al., 2021; Rathore et al., 2024). Prior work addresses these challenges through adaptive activations, Fourier features, adaptive loss weighting, exact boundary constraints, gradient-enhanced objectives, causal training, and domain-decomposition or variational formulations (Jagtap et al., 2020; Wang et al., 2021b; McClenny & Braga-Neto, 2023; Sukumar & Srivastava, 2022; Yu et al., 2022; Wang et al., 2024; Jagtap & Karniadakis, 2020; Kharazmi et al., 2021). These advances highlight the multi-component nature of effective PINN design, but individual methods typically target particular training difficulties specified in advance.

Automated approaches instead search for suitable PINN designs. Auto-PINN and NAS-PINN automate architecture and hyperparameter selection (Wang et al., 2023; Wang & Zhong, 2024), while evolutionary methods optimize architectures, activation functions, loss weights, and related hyperparameters (Kaplarević-Mališić et al., 2023; Zhang & Yang, 2024; Carrillo et al., 2024). More recently, PINNsAgent uses LLM-based knowledge replay and memory-guided reasoning to search PINN configurations, whereas Lang-PINN employs multiple agents for PDE understanding, code generation, execution, and correction (Wuwu et al., 2025; You et al., 2026). PINNForge differs by treating execution evidence from real training as a persistent cross-generation design signal and by evolving complete, coupled PINN configurations through semantic inheritance, revision, recombination, and exploration rather than predefined parameter-level operators.

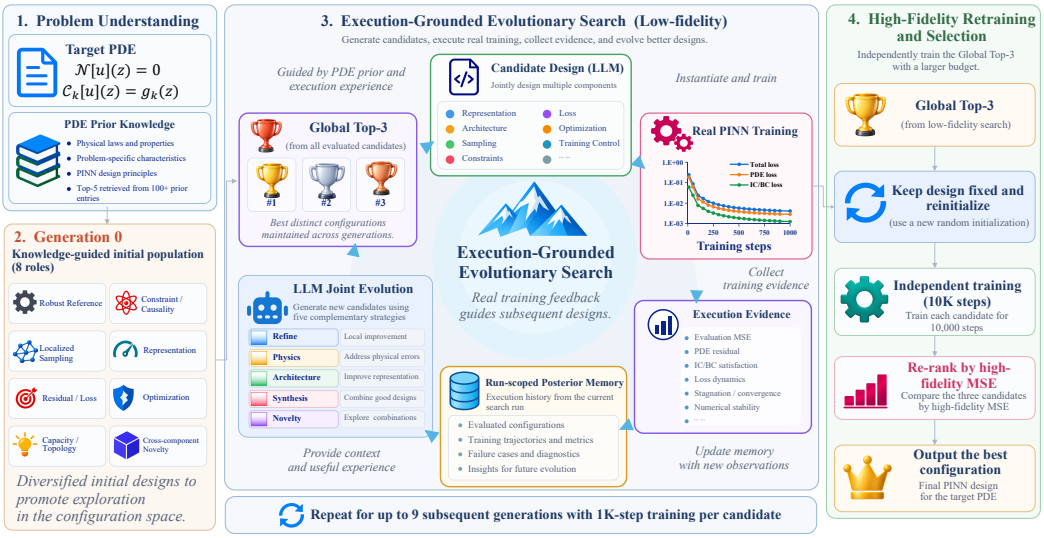


Figure 1: Overview of the PINNForge framework.

2.2 LLM-Guided Evolutionary Algorithm Discovery

LLMs are increasingly used within iterative algorithm-discovery systems. FunSearch repeatedly evaluates generated programs and reuses strong solutions as context (Romera-Paredes et al., 2024). EoH integrates LLM-based generation with evolutionary search in a generate–execute–evaluate–evolve loop (Liu et al., 2024; Yao et al., 2025; Liu et al., 2026). ReEvo translates performance differences into language-based reflective feedback (Ye et al., 2024), while LLaMEA couples runtime evaluation, selection, and iterative modification for metaheuristic design (van Stein & Bäck, 2025; van Stein et al., 2025). Together, these studies show that LLMs can support semantic evolutionary operations by deciding which mechanisms to preserve, revise, combine, or replace based on candidate structure and execution behavior. PINNForge extends this principle to PINN configuration search, where each execution yields not only a scalar performance measure but also physically meaningful residual, constraint, and optimization evidence.

3 Method

3.1 Overview

PINNForge adapts the generate–execute–evaluate–evolve principle of Evolution of Heuristics (EoH) to automated PINN design. Rather than generating a configuration once, it repeatedly trains candidate designs on the target PDE and uses the resulting performance, physical, and optimization evidence to guide subsequent generations. The search is therefore grounded in observed PDE-solving behavior rather than solely in static PDE descriptions or predefined parameter-level perturbations.

Given a target PDE, PINNForge combines problem characteristics with relevant PINN prior knowledge to generate a diverse initial population. Each candidate is trained to obtain a scalar performance measure together with diagnostic execution evidence. The best distinct configurations over the full search history form the Global Top-3 and serve as parents for subsequent generations. The LLM then uses these elites, their execution evidence, and accumulated run-level experience to refine, recombine, or explore beyond existing designs. After low-fidelity search terminates, the final Global Top-3 configurations are independently retrained under a larger budget, and the best-performing high-fidelity candidate is returned. Figure 1 summarizes the overall framework.

3.2 EXECUTION-GROUNDED SEARCH CONTEXT

Let z denote the input coordinates and $u(z) \in \mathbb{R}^{d_u}$ denote the physical field governed by a target PDE $\mathcal{P}$. We express the governing equation and associated physical constraints as

$$\mathcal{N}[u](z) = 0, \qquad z \in \mathcal{D},$$

and

$$\mathcal{C}_k[u](z) = g_k(z), \qquad z \in \Gamma_k,$$

respectively. A PINN $u_\theta(z)$ approximates the physical field by minimizing

$$\mathcal{L}(\theta) = \lambda_r \mathcal{L}_r + \sum_k \lambda_k \mathcal{L}_k,$$

where $\mathcal{L}_r$ denotes the PDE residual loss and $\mathcal{L}_k$ denotes the loss associated with the k-th initial, boundary, or other physical constraint. Candidate accuracy is evaluated by mean squared error (MSE) on fixed evaluation points.

PINNForge conditions candidate generation on two complementary sources of information. The first is PDE-related prior knowledge. PINNForge maintains a fixed prior-knowledge base $\mathcal{K}$ containing more than one hundred entries on general physical reasoning and PINN design principles. For a target PDE $\mathcal{P}$, the method identifies properties relevant to PINN design—including time dependence, periodicity, multiscale structure, geometric complexity, locally steep behavior, and physical-constraint types—and retrieves the five most relevant entries from $\mathcal{K}$. These entries form the PDE-specific prior context $K_{\mathcal{P}}$, which provides design rationale rather than prescribing a fixed solver configuration and remains available throughout the search.

The second source is execution evidence obtained from candidates trained on the current PDE. For a candidate configuration a_i, we denote its execution as

$$\mathrm{Execute}(a_i; \mathcal{P}) \to (y_i, E_i),$$

where y_i is the scalar comparison metric used for ranking, and E_i records the physical and optimization diagnostics observed during training. Across candidates, these records capture PDE residuals, constraint satisfaction, convergence behavior, and numerical stability, providing diagnostic information about why a design succeeds or fails.

PINNForge uses execution evidence at two granularities. Selected parents retain candidate-specific evidence $E_{P_{g+1}}$, while the full history through Generation g is compressed into a run-scoped posterior memory M_g. The former preserves detailed evidence associated with specific parent configurations, whereas the latter summarizes recurring successes, failures, and design experience across the current search run. Appendix A.3 details the evidence fields and the construction of M_g.

3.3 LLM-DRIVEN EVOLUTIONARY PINN DESIGN

Each candidate is represented as a structured `AlgorithmSpec` spanning representation, architecture, sampling, physical constraints, loss construction, optimization, and training control. The LLM proposes specifications rather than arbitrary executable code, while a deterministic controller validates and maps compatible, budget-compliant specifications to supported PINN implementations. Appendix A.1 summarizes the executable design dimensions and representative choices.

At Generation 0, no execution evidence from the current PDE is available. PINNForge therefore generates eight complete initial configurations under complementary design roles that emphasize different potential bottlenecks without prescribing fixed templates. Appendix A.4 defines these roles.

After candidate execution, PINNForge performs global elite selection over the full search history. Let $\mathcal{H}_g$ denote all candidates evaluated through Generation g. The parent set for the next generation is

$$P_{g+1} = \mathrm{GlobalDistinctTop3}\left(\mathcal{H}_g\right).$$

Selecting the three best distinct configurations over the full search history preserves strong early designs and provides multiple validated parents for subsequent refinement and recombination.

From Generation 1 onward, new candidates are conditioned on the Global Top-3, their candidate-specific execution evidence, PDE-related prior knowledge, and the run-scoped posterior memory.

Because PINN components are coupled, PINNForge may revise several related components jointly rather than relying on independent numerical perturbations.

Each generation employs five complementary strategies. Refine makes conservative adjustments around the strongest design; Physics-Guided targets sampling, constraints, and loss construction; Architecture-Guided focuses on representation, architecture, and optimization; Synthesis combines complementary mechanisms from multiple strong parents; and Novelty explores beyond the current elite trajectory. Appendix A.4 provides the full definitions.

For strategy s, a next-generation candidate is generated as

$$a_{g+1}^{(s)} = \text{LLMGenerate}\left(P_{g+1}, E_{P_{g+1}}, K_{\mathcal{P}}, M_g, s\right),$$

where $E_{P_{g+1}}$ denotes the selected parents' candidate-specific execution evidence, $K_{\mathcal{P}}$ the PDE-related prior knowledge, and M_g the run-scoped posterior memory summarized from the search history through Generation g.

The strategy specifies the search direction, while the LLM decides which mechanisms to inherit, revise, recombine, or explore based on observed execution behavior. Executing the new candidates expands the history from $\mathcal{H}_g$ to $\mathcal{H}_{g+1}$, after which the posterior-summary step updates M_g to M_{g+1}. The new candidates also compete with all previously evaluated candidates for the Global Top-3, closing the generate–execute–evaluate–evolve loop.

3.4 Multi-Fidelity Search and High-Fidelity Evaluation

Evaluating many candidates at the full training budget would be expensive. Inspired by multi-fidelity neural architecture search (Phan & Luong, 2024; Won et al., 2025), PINNForge separates low-fidelity exploration from high-fidelity evaluation.

During low-fidelity search, every valid candidate receives the same $1{,}000$-step training budget, enabling efficient comparison while still producing execution evidence. Generation 0 contains eight candidates, followed by at most nine subsequent generations with five candidates each. The Global Top-3 is updated over the full search history.

Low-budget rankings may not perfectly predict fully trained performance (Yu et al., 2020). PINNForge therefore does not directly return the best low-fidelity candidate. Once low-fidelity search terminates, the final Global Top-3 configurations are kept fixed, independently reinitialized, and retrained from scratch for $10{,}000$ optimizer steps each. The candidate with the lowest high-fidelity evaluation MSE on the fixed reference points is then selected.

This two-stage procedure uses low-fidelity training for broad exploration and independent high-fidelity retraining to reduce dependence on transient low-budget rankings and particular parameter initializations.

4 Experiments

We evaluate PINNForge on 25 heterogeneous PDE benchmarks, focusing on solution accuracy, the roles of its main search mechanisms, and performance evolution across generations.

4.1 Experimental Setup

Of the 25 PDEs, 21 are drawn from PINNacle (Hao et al., 2024), spanning nonlinear transport, reaction–diffusion, multiscale and long-time dynamics, complex geometries, high-dimensional PDEs, inverse problems, and fluid systems. We additionally include Allen–Cahn, variable-coefficient Darcy flow, the shallow-water system, and Kovasznay flow (Allen & Cahn, 1979; Darcy, 1856; de Saint-Venant, 1871; Kovasznay, 1948). Definitions of all PDEs are provided in Appendix D. We compare PINNForge with RoPINN, PINNsFormer, and PINNsAgent (Wu et al., 2024; Zhao et al., 2024; Wuwu et al., 2025), representing PINN optimization, architectural design, and LLM-based automated configuration, respectively. Qwen3.8-27B is used as the fixed LLM backbone without PDE-specific fine-tuning.

For each PDE, solution accuracy is measured by MSE on fixed reference evaluation points. The same points are used for candidate comparison, search control, final configuration selection, and reporting;

Table 1: Mean ± standard deviation of MSE over 10 independent runs on 25 PDEs, computed on the fixed reference evaluation points. Lower is better.

PDE	RoPINN	PINNsFormer	PINNsAgent	PINNForge
burgers_1d	7.31E-04±6.13E-05	1.08E-03±8.21E-06	6.52E-05±1.55E-05	**6.50E-05±4.42E-07**
burgers_2d	2.40E-01±3.12E-02	2.65E-01±6.83E-02	1.98E-01±1.66E-02	**1.95E-01±1.31E-02**
gray_scott_2d	8.64E-03±4.53E-05	1.57E-02±1.32E-03	4.36E-03±3.08E-05	**3.60E-03±2.65E-04**
heat_2d_complex_geometry	**4.00E-04±9.10E-05**	2.78E-03±5.77E-05	1.84E-03±9.95E-04	4.45E-04±9.72E-05
heat_2d_long_time	2.56E+00±4.53E-01	3.83E+00±1.64E-01	4.32E+00±2.78E+00	**1.13E+00±4.21E-02**
heat_2d_multiscale	2.75E-05±7.52E-06	5.57E-05±9.93E-06	3.51E-05±2.20E-05	**1.19E-05±2.16E-06**
heat_2d_varying_coefficient	6.24E-01±4.53E-03	1.19E-02±3.42E-03	5.34E-03±3.85E-03	**1.59E-03±4.77E-04**
heat_5d	5.39E-07±1.31E-07	1.87E-07±1.91E-08	3.34E-07±7.65E-08	**3.21E-08±3.26E-08**
heat_inverse_2d	4.31E-02±7.31E-03	8.89E-02±4.22E-02	6.32E-02±8.63E-02	**7.15E-03±4.91E-03**
kuramoto_sivashinsky_1d	6.54E+00±5.14E-01	5.13E+00±4.51E-03	1.09E+00±3.18E-02	**1.03E+00±8.40E-02**
navier_stokes_2d_backstep	9.64E-03±5.13E-03	6.02E-04±4.33E-05	4.96E-03±1.45E-03	**2.63E-04±4.71E-05**
navier_stokes_2d_classic	2.88E-05±3.61E-06	1.49E-05±1.26E-05	8.48E-06±6.90E-06	**2.48E-06±2.31E-06**
navier_stokes_2d_long_time	5.43E+02±5.63E+00	7.43E+02±1.31E+02	5.75E+02±7.63E+00	**5.04E+02±4.95E+00**
poisson_2d_classic	8.57E-02±6.74E-02	2.17E-01±7.65E-02	6.34E-01±1.43E-01	**1.61E-02±3.95E-04**
poisson_boltzmann_2d	7.91E-01±7.14E-01	1.34E-02±4.31E-03	6.97E-02±7.43E-02	**5.38E-03±9.13E-04**
poisson_2d_many_area	2.89E+00±3.51E-01	5.36E+00±3.79E+00	3.00E+00±1.01E+00	**2.21E+00±3.56E-01**
poisson_3d_complex_geometry	2.16E-01±1.53E-02	8.16E-03±5.58E-04	1.60E-02±1.08E-02	**1.04E-03±2.34E-04**
poisson_5d	4.09E-07±7.43E-08	2.92E-07±1.99E-07	2.01E-06±1.09E-06	**5.64E-08±3.95E-08**
poisson_inverse_2d	5.32E-03±3.46E-03	1.51E-03±8.65E-04	2.41E-02±4.73E-02	**2.57E-04±1.29E-04**
wave_1d	4.19E-02±4.38E-03	3.61E-02±6.65E-03	3.25E-02±3.45E-02	**3.18E-02±5.73E-03**
wave_2d_heterogeneous	4.63E-02±6.73E-03	8.37E-02±1.18E-02	5.17E-02±8.03E-03	**2.84E-02±7.69E-04**
allen_cahn_1d	7.45E-08±4.61E-08	6.02E-09±1.08E-09	1.44E-08±3.50E-09	**1.13E-09±2.26E-10**
darcy_flow_2d	3.48E-09±1.63E-09	1.35E-10±2.90E-11	5.33E-07±2.65E-07	**1.64E-11±3.28E-12**
shallow_water_2d	5.78E-07±7.13E-07	2.24E-07±8.44E-08	6.81E-07±1.25E-07	**7.15E-08±2.86E-08**
kovasznay_flow_2d	6.13E-07±4.31E-07	3.80E-08±9.63E-09	3.82E-07±3.66E-07	**1.77E-09±4.43E-10**
Friedman Avg. Rank	3.04	2.96	2.96	**1.04**
Wilcoxon vs. PINNForge	+	+	+	–
Median CV	0.177	0.184	0.497	0.182

The Friedman test gives $\chi_F^2 = 42.696$ and $p = 2.86 \times 10^{-9}$. For Wilcoxon tests, "+" indicates that PINNForge significantly outperforms the corresponding baseline under paired signed-rank tests with Holm correction ($p_{\mathrm{adj}} = 1.79 \times 10^{-7}$ for each comparison). CV denotes the median coefficient of variation across the 25 PDEs.

no separate held-out reference subset is introduced. We report mean MSE over 10 independent runs, each corresponding to one complete execution of the method, including its internal search and selection procedure. PINNForge trains each low-fidelity candidate for $1{,}000$ optimizer steps and independently retrains each configuration in the final Global Top-3 for $10{,}000$ steps before selecting the candidate with the lowest high-fidelity MSE. Appendix B.2–B.4 provides details on LLM sampling, the computational environment, and the evaluation protocol.

Table 1 reports mean±standard-deviation MSE over 10 independent runs on all 25 heterogeneous PDEs, together with aggregate statistical results. Appendix B.6 provides per-task ranks, the PINNacle oracle reference, and additional statistical details.

PINNForge achieves the lowest mean MSE among the four compared methods on 24 of 25 PDEs; RoPINN is slightly better only on `heat_2d_complex_geometry`. Relative to the strongest baseline on each task, PINNForge yields a median MSE reduction of 59.85%, with a geometric-mean best-baseline-to-PINNForge MSE ratio of $2.78\times$. It also ranks first on all four PDEs added beyond PINNacle, indicating that the gains are not confined to a single equation family within this benchmark.

As summarized in Table 1, across the 25 PDEs, PINNForge achieves an average Friedman rank of 1.04, compared with 3.04, 2.96, and 2.96 for RoPINN, PINNsFormer, and PINNsAgent, respectively. A Friedman test on per-task ranks computed from the 10-run mean MSE indicates significant overall differences ($\chi_F^2 = 42.696$, $p = 2.86 \times 10^{-9}$). Pairwise Wilcoxon signed-rank tests on the $\log_{10}$-transformed per-task mean MSE across the 25 tasks, followed by Holm correction, favor PINNForge over all three baselines ($p_{\mathrm{adj}} = 1.79 \times 10^{-7}$ for each comparison). Median coefficients of variation are 0.177, 0.184, 0.497, and 0.182 for RoPINN, PINNsFormer, PINNsAgent, and PINNForge, respectively.

The magnitude of improvement remains task-dependent. Relative to the strongest baseline, PINNForge achieves $3.36\times$ lower MSE on `heat_2d_varying_coefficient`, $5.83\times$ on `heat_5d`, $7.85\times$ on `poisson_3d_complex_geometry`, $8.23\times$ on `darcy_flow_2d`, and $21.47\times$ on `kovasznay_flow_2d`, whereas the margins are small on `burgers_1d`, `burgers_2d`, and `wave_1d`. This pattern is consistent with greater benefits on tasks that require coordinated choices across multiple PINN components, although these task-wise comparisons do not isolate which problem properties drive the gains.

Table 2: Ablation summary over 12 representative PDEs. The last column reports the median task-wise MSE ratio relative to the full PINNForge; lower is better.

Variant	Knowl. guidance	Exec. feedback	Parent evolution	Median ratio
w/o Knowledge Guidance	×	✓	✓	3.74×
w/o Execution Feedback to LLM	✓	×	✓	12.10×
w/o Evolutionary Search	✓	✓	×	10.10×
PINNForge	✓	✓	✓	1.00×

Table 3: PINN optimizer-step budget used to obtain the selected result for one PDE.

Method	Selection protocol	Budget
RoPINN	Best of 5 independently initialized 20K-step candidates	100K
PINNsFormer	Best of 5 independently initialized 20K-step candidates	100K
PINNsAgent	Best of 5 independently initialized 20K-step candidates	100K
PINNForge	53×1K LF + Top-3×10K HF	≤83K

4.2 Component Ablation

We evaluate three ablations on 12 representative PDEs to examine the roles of knowledge guidance, execution feedback, and parent-driven evolution. Knowledge guidance comprises the PDE prior $K_{\mathcal{P}}$ and run-scoped posterior memory M_g; `w/o Knowledge Guidance` removes both while retaining candidate-specific execution evidence and parent-driven evolution. `w/o Execution Feedback` withholds execution evidence from the LLM and consequently disables the execution-derived posterior memory, while preserving PDE-related prior knowledge and parent-driven evolution. `w/o Evolutionary Search` keeps the candidate-evaluation budget fixed but replaces parent-driven evolution with independent LLM generation. Table 2 summarizes these interventions and their aggregate effects. Appendix A.6 details the information available to each variant, and Appendix B.7 reports the complete per-task results.

Removing Execution Feedback produces the largest aggregate degradation, increasing the median MSE to $12.10\times$ that of full PINNForge. Because this variant withholds candidate-specific execution evidence and therefore cannot form the execution-derived posterior memory, the remaining PDE-related prior knowledge and parent structures are insufficient to recover full-method performance under this setup. Removing Evolutionary Search increases the median error to $10.10\times$ under the same candidate-evaluation budget, consistent with a benefit from parent-driven inheritance and recombination. Removing Knowledge Guidance increases the median error to $3.74\times$; because both $K_{\mathcal{P}}$ and M_g are removed, this result reflects the knowledge-guidance channel as a whole rather than static prior knowledge alone.

Per-task results reveal distinct degradation patterns. On `heat_2d_complex_geometry`, removing Execution Feedback increases MSE from 4.45×10^{-4} to 1.59, an increase of roughly 3.6×10^3. Removing parent-driven evolution produces particularly large increases on `poisson_5d` and `darcy_flow_2d`, where MSE rises by about $365\times$ and $652\times$, respectively. These cases are consistent with complementary roles for diagnostic feedback and inheritance, but do not constitute isolated causal attributions because multiple design components can change simultaneously.

4.3 Training Budget

Table 3 compares the optimizer-step budget required to obtain the selected result for a single PDE. Each of the three baselines trains five independently initialized candidates for $20{,}000$ optimizer steps and selects the one with the lowest evaluation MSE, resulting in a total budget of $100{,}000$ steps. PINNForge instead uses at most $53{,}000$ steps for low-fidelity search and $30{,}000$ steps for independent high-fidelity retraining of the final Global Top-3, yielding a maximum total of $83{,}000$ optimizer steps.

The reported gains therefore do not rely on a larger PINN optimizer-step budget than the baselines. This accounting compares optimization steps rather than total computational cost, since per-step cost can vary with network architecture, sampling scale, and optimizer configuration. It also excludes LLM inference, wall-clock time, and monetary cost; Appendix B.8 provides the full accounting.

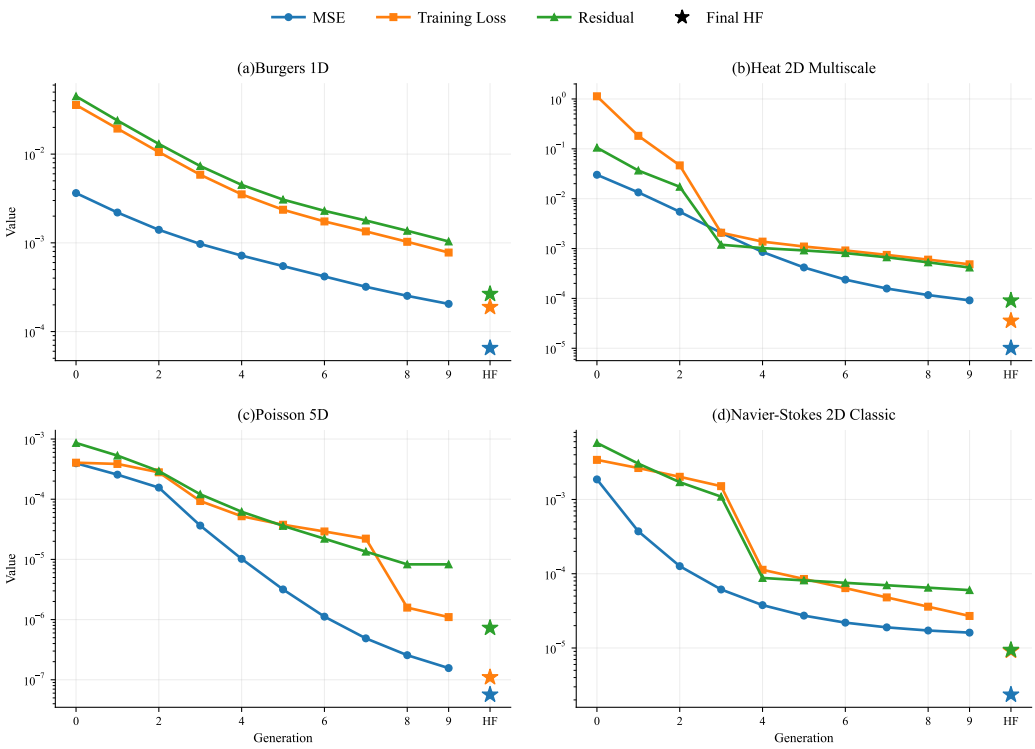


Figure 2: Search trajectories on four representative PDEs. At each generation, the candidate with the lowest low-fidelity MSE is selected, and its MSE, training loss, and PDE residual are shown. Stars mark the corresponding results after independent high-fidelity retraining of the final Global Top-3.

Table 4: Representative single-run trajectories. G0 is the best Generation-0 evaluation MSE, LF the best low-fidelity MSE, and HF the best MSE after high-fidelity retraining.

PDE	G0	Best LF	Final HF	G0→HF	LF→HF
burgers_1d	3.64E-03	2.05E-04	6.52E-05	55.81×	3.15×
heat_2d_multiscale	3.02E-02	9.14E-05	1.01E-05	2,986.95×	9.05×
poisson_5d	3.96E-04	1.56E-07	5.65E-08	7,008.58×	2.77×
navier_stokes_2d_classic	1.86E-03	1.61E-05	2.35E-06	792.89×	6.85×

4.4 Search Process Analysis

We next examine whether performance continues to improve beyond Generation 0 on four representative PDEs spanning nonlinear transport, multiscale dynamics, high-dimensional PDEs, and fluid systems. These trajectories are obtained from separate single-run searches conducted specifically for process analysis and are independent of the ten-run experiments used for aggregate performance statistics. Figure 2 tracks the candidate with the lowest low-fidelity MSE in each generation, together with its corresponding training loss and PDE residual. Stars mark the corresponding results after independent high-fidelity retraining of the final Global Top-3. Appendix C reports the corresponding trajectories for all 25 PDEs.

All four representative searches discover a new low-fidelity global best after Generation 0. The training loss and PDE residual associated with the generation-best MSE candidates also decrease broadly across the four searches, indicating that the reduction in evaluation error is generally accompanied by improvements in the training objective and PDE consistency. As summarized in Table 4, low-fidelity evolution reduces MSE by $17.76\times$–$2{,}538.46\times$ relative to the best Generation-0 candidate, while high-fidelity retraining increases the total improvement to $55.81\times$–$7{,}008.58\times$.

The Burgers-1D trajectory illustrates how execution evidence informs subsequent design decisions. For parent `G6-C4`, training exhibited stable optimization and strong hard-constraint satisfaction but a persistent residual hotspot near the shock. Child `G7-C4` therefore preserved the stable architecture and constraint mechanisms while strengthening local adaptive sampling and revising the optimization configuration; solution MSE decreased from 5.71×10^{-4} to 2.21×10^{-4}, while the PDE residual decreased from 6.05×10^{-2} to 2.14×10^{-2}. Because the child is independently initialized and modifies both sampling and optimization, this trace does not attribute the improvement to any single component. Appendix A.7 provides the complete component-level trace and evolutionary lineage.

The additional $2.77\times$–$9.05\times$ improvement from Best LF to Final HF shows that configurations identified under the low-fidelity budget can improve further under longer independent optimization. Together with the generation-wise gains, these trajectories provide evidence that performance improvements continue beyond the initial population, while not isolating the causal contribution of any individual search mechanism.

4.5 Discussion

These analyses provide complementary evidence. The 25-PDE benchmark establishes the overall accuracy gains, the budget-matched ablations show that candidate-evaluation budget alone does not account for the advantage, the search trajectories demonstrate continued improvement beyond Generation 0, and the Burgers trace illustrates how execution evidence informs subsequent design decisions. Together, these results are consistent with a beneficial interaction between execution evidence and parent-driven evolution over historical elites, although they do not isolate the causal contribution of individual mechanisms.

The gains remain task-dependent, with RoPINN slightly outperforming PINNForge on `heat_2d_complex_geometry`. Because the same fixed reference points are reused for search control, final selection, and reporting, the reported MSE reflects performance under this benchmark protocol rather than held-out generalization, and adaptive reuse may favor configurations tuned to these points. The study also uses a single LLM backbone, Qwen3.8-27B, and compares PINN optimizer-step budgets rather than end-to-end computational cost. Cross-backbone robustness, independently sampled evaluation points, and end-to-end efficiency therefore remain important directions for further evaluation.

5 Conclusion

We developed PINNForge, an execution-grounded evolutionary framework for automated PINN design that integrates PDE-related prior knowledge, historical elites, execution evidence, and LLM-driven cross-component evolution. Rather than treating candidate training as a terminal evaluation step, PINNForge feeds observed performance, physical, and optimization evidence into subsequent generations, enabling complete PINN configurations to be iteratively revised, recombined, and explored within a closed loop.

Across 25 heterogeneous PDEs, PINNForge achieves the lowest mean MSE on 24 tasks with an average Friedman rank of 1.04, while using at most $83{,}000$ PINN optimizer steps per run. Controlled ablations show substantial degradation when knowledge guidance, execution feedback, or parent-driven evolution is removed, while search trajectories demonstrate that strong configurations continue to emerge beyond Generation 0 and can improve further under independent high-fidelity retraining. Together, these results support execution-grounded cross-generation evolution as a promising direction for moving automated PINN design beyond one-shot configuration generation toward iterative design improvement driven by observed PDE-solving behavior.

AI USE STATEMENT

Qwen3.8-27B was used as a component of PINNForge to generate candidate PINN configurations and refine subsequent candidates using feedback from PINN training. The generated configurations were validated and evaluated through PINN training. ChatGPT was used for language editing and polishing of the manuscript. The authors reviewed the AI-assisted work and take full responsibility for the methodology, experiments, results, and final content.

REPRODUCIBILITY STATEMENT

We provide an anonymous implementation at the link given in the abstract. Appendix A specifies the structured candidate representation, LLM context, initialization roles, evolutionary strategies, and controller-side validation. Appendix B provides the search configuration, LLM sampling settings, computational environment, evaluation protocol, complete results, and training-budget accounting. Appendix C reports complete search trajectories, and Appendix D gives the definitions of all benchmark PDEs used in the experiments.

REFERENCES

Samuel M Allen and John W Cahn. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta metallurgica*, 27(6):1085–1095, 1979.

Jinshuai Bai, Yizheng Wang, Hyogu Jeong, Shiyuan Chu, Qingxia Wang, Laith Alzubaidi, Xiaoying Zhuang, Timon Rabczuk, Yi Min Xie, Xi-Qiao Feng, et al. Towards the future of physics-and data-guided AI frameworks in computational mechanics. *Acta Mechanica Sinica*, 41(7):225340, 2025.

Hugo Carrillo, Taco de Wolff, Luis Martí, and Nayat Sanchez-Pi. Evolutionary multi-objective physics-informed neural networks: The MOPINNs approach. *AI Communications*, 37(3):397–409, 2024.

Henry Darcy. *Les fontaines publiques de la ville de Dijon: Exposition et application des principes à suivre et des formules à employer dans les questions de distribution d'eau*. Victor Dalmont, Paris, 1856.

Adhémar Jean-Claude de Saint-Venant. Théorie du mouvement non-permanent des eaux, avec application aux crues des rivières et à l'introduction des marées dans leur lit. *CR Acad. Sci. Paris*, 73:147–154, 237–240, 1871.

Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, and Jun Zhu. PINNacle: A comprehensive benchmark of physics-informed neural networks for solving PDEs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 76721–76774. Curran Associates, Inc., 2024. doi: 10.52202/079017-2442.

Ameya D. Jagtap and George Em Karniadakis. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5): 2002–2041, 2020. doi: 10.4208/cicp.OA-2020-0164.

Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

Ana Kaplarević-Mališić, Branka Andrijević, Filip Bojović, Srđan Nikolić, Lazar Krstić, Boban Stojanović, and Miloš Ivanović. Identifying optimal architectures of physics-informed neural networks by evolutionary strategy. *Applied Soft Computing*, 146:110646, 2023.

Ehsan Kharazmi, Zhongqiang Zhang, and George E. M. Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021. doi: 10.1016/j.cma.2020.113547.

L. I. G. Kovasznay. Laminar flow behind a two-dimensional grid. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44(1):58–62, 1948. doi: 10.1017/S0305004100023999.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in neural information processing systems*, volume 34, pp. 26548–26560, 2021.

Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223. PMLR, 2024.

Fei Liu, Yilu Liu, Qingfu Zhang, Tong Xialiang, and Mingxuan Yuan. EoH-S: Evolution of heuristic set using LLMs for automated heuristic design. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(43):37090–37098, 2026. doi: 10.1609/aaai.v40i43.41038.

Levi D. McClenny and Ulisses M. Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023. doi: 10.1016/j.jcp.2022.111722.

Alkmini Michaloglou, Ioannis Papadimitriou, Ilias Gialampoukidis, Stefanos Vrochidis, and Ioannis Kompatsiaris. Physics-informed neural networks in materials modeling and design: A review. *Archives of Computational Methods in Engineering*, 33(4):5223–5260, 2026.

Quan Minh Phan and Ngoc Hoang Luong. Efficient multi-fidelity neural architecture search with zero-cost proxy-guided local search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 232–240, 2024.

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. doi: 10.1016/j.jcp.2018.10.045.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training PINNs: A loss landscape perspective. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42159–42191. PMLR, 21–27 Jul 2024.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Natarajan Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

Juan Diego Toscano, Vivek Oommen, Alan John Varghese, Zongren Zou, Nazanin Ahmadi Daryakenari, Chenxi Wu, and George Em Karniadakis. From PINNs to PIKANs: Recent advances in physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, 1(1):15, 2025.

Niki van Stein and Thomas Bäck. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29(2): 331–345, 2025. doi: 10.1109/TEVC.2024.3497793.

Niki van Stein, Diederick Vermetten, and Thomas Bäck. In-the-loop hyper-parameter optimization for LLM-based automated design of heuristics. *ACM Transactions on Evolutionary Learning and Optimization*, 2025. doi: 10.1145/3731567.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a. doi: 10.1137/20M1318043.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021b. doi: 10.1016/j.cma.2021.113938.

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, 2022. doi: 10.1016/j.jcp.2021.110768.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024. doi: 10.1016/j.cma.2024.116813.

Yicheng Wang, Xiaotian Han, Chia-Yuan Chang, Daochen Zha, Ulisses Braga-Neto, and Xia Hu. Auto-PINN: Understanding and optimizing physics-informed neural architecture. In *NeurIPS 2023 Workshop on AI for Science*, 2023.

Yifan Wang and Linlin Zhong. NAS-PINN: Neural architecture search-guided physics-informed neural network for solving PDEs. *Journal of Computational Physics*, 496:112603, 2024. doi: 10.1016/j.jcp.2023.112603.

Jonghyeon Won, Hyun-Suk Lee, and Jang-Won Lee. A review on multi-fidelity hyperparameter optimization in machine learning. *ICT Express*, 11(2):245–257, 2025.

Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023. doi: 10.1016/j.cma.2022.115671.

Haixu Wu, Huakun Luo, Yuezhou Ma, Jianmin Wang, and Mingsheng Long. RoPINN: Region optimized physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 110494–110532. Curran Associates, Inc., 2024. doi: 10.52202/079017-3508.

Qingpo Wuwu, Chonghan Gao, Tianyu Chen, Yihang Huang, Yuekai Zhang, Jianing Wang, Jianxin Li, Haoyi Zhou, and Shanghang Zhang. PINNsAgent: Automated PDE surrogation with large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68143–68165. PMLR, 2025.

Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. Multi-objective evolution of heuristic using large language model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25):27144–27152, 2025.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems*, volume 37, pp. 43571–43608. Curran Associates, Inc., 2024. doi: 10.52202/079017-1381.

Liangliang You, Xin He, Hongduan Tian, Bo Han, Ivor Tsang, and Yew-Soon Ong. Lang-PINN: From language to physics-informed neural networks via a multi-agent framework. In *ICLR 2026 Workshop on AI with Recursive Self-Improvement*, pp. 1–35, 2026.

Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2020.

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, 2024.

Zhiyuan Zhao, Xueying Ding, and B. Aditya Prakash. PINNsFormer: A transformer-based framework for physics-informed neural networks. In *International Conference on Learning Representations*, pp. 38475–38491, 2024.