PINNeval: A Comprehensive Evaluation Standard for Physics-Informed Neural Networks

NeurIPS 2026 · Evaluations & Datasets Track

Kevin von Bargen · Hamid Ebrahimy · Tim Römer
Joint Lab Artificial Intelligence and Data Science, Osnabrück University, Osnabrück, Germany · kvonbargen@uos.de

Four-component decomposition of the PINN training pipeline

Domain-importance adaptation decides where residuals are evaluated and how points are weighted; loss balancing weights whole equations. They are meant to be used jointly, yet earlier benchmarks such as PINNacle treat them as one category and compare them against each other.

Optimization scheme
optimizer(s) and stages that minimize over θ

Domain-importance adaptation
collocation sets C_e and pointwise weights ω(t,x)

$$\min_{\theta} \sum_{e \in \mathcal{E}} \lambda_e \frac{1}{|C_e|} \sum_{(t,x) \in C_e} \omega(t,x)\, r_e(t,x;\theta)^2$$

Loss balancing
equation-level weights λ_e

Neural architecture
the network u_θ inside every residual r_e

PINNeval's three principles

Competitive

Components not under study are competitive, not a minimal default.

Coupled

Results are conditional on the surrounding pipeline: report all four.

Costed

Runtime and hardware are reported next to the iteration counts.

PINNeval: ten items to specify, control and report in every study

Per component

1	**Model specification**	topology, depth, width, activations, features
2	**Optimization scheme**	optimizers, schedule, stopping, budget
3	**Loss balancing**	method for λ_e, update frequency
4	**Domain-importance adaptation**	method, set sizes, update frequency

Per evaluation

5	**Claim under evaluation**	stated, with the PDE problems
6	**Comparison conditions**	baselines, fixed parts, competitiveness
7	**Metrics**	relative L^2 error; runtime + iterations
8	**Numerical precision**	float32 / float64, reported
9	**Robustness & statistics**	seeds, variability, failures, test
10	**Reproducibility & limitations**	all settings; where it does not hold

EvalCard: Domain-importance adaptation

Configuration.
varied | GradNorm-based | MLP | SOAP

Tested.
vRBA had not been compared with a representative set of DIA methods; LAS had not been evaluated inside a competitive pipeline.

Revealed.
Random-R is hard to beat: best median on 7 of 9 benchmarks at the lowest cost. LAS's benefit transfers, with small, inconsistent margins.

EvalCard: Neural network architecture

Configuration.
Random-R | GradNorm-based | **varied** | SOAP

Tested.
Once the surrounding pipeline is fixed to a strong configuration, the choice of architecture has a measurable and consistent influence.

Revealed.
Supported only weakly: no architecture dominates across benchmarks and sizes. At large scale the MLP remains competitive or superior and is substantially cheaper to train.

EvalCard: Loss balancing

Configuration.
Random-R | **varied** | MLP | SOAP

Tested.
ReLoBRaLo's comparison omits GradNorm-based balancing.

Revealed.
No method dominates. GradNorm-based balancing has the lowest median error on 5 of 9 benchmarks, fixed weights on 3. Robustness failures: SoftAdapt (Wave, KS) and GradNorm-MTL (KdV).

EvalCard: Optimizer & network

Configuration.
LAS | GradNorm-based | **varied** | **varied**

Tested.
Wang et al. (NeurIPS 2025) exclude quasi-Newton methods, which typically need full-batch training, float64 and small networks: "impractical for the complex PDE benchmarks considered in this work."

Revealed.
Inside a competitive pipeline, a two-stage SOAP → SSBroyden2 scheme on a moderate-size MLP in float64 is faster on all 9 benchmarks and has a significantly lower error on 7 (RT, GL: not significant).
The claim is revised.

PINNpipe – the JAX library

Domain-importance adaptation (6)
Random-R | grid | RAD | LAS | R3 | vRBA | **+ your method**

Loss balancing (6)
fixed | GradNorm-based | LR annealing | ReLoBRaLo | SoftAdapt | GradNorm-MTL | **+ your method**

Neural architecture (3)
MLP | Modified MLP | PirateNet | **+ your method**

Optimization scheme (3)
Adam | SOAP | SSBroyden | **+ your method**

Benchmarks Wave | Burgers | Allen–Cahn | KdV | Kuramoto–Sivashinsky | Ginzburg–Landau | Rayleigh–Taylor | Heat 5D | Poisson 5D

Use it: access & documentation

Paper: openreview.net/forum?id=O6VplBRoF0
Code: github.com/KevinvonBargen/pinnpipe (JAX, Apache-2.0)
PINNpipe: one self-contained module per component with a uniform interface; 18 methods and nine benchmark problems.

Document an evaluation with the EvalCard template in the paper's appendix.
Data: reference data and reference-solution scripts ship with PINNpipe.
Scope and limits: PINNs and forward problems; no complex geometries.

Paper | Code

Supported by the Lower Saxony Ministry of Science and Culture (MWK) through the zukunft.niedersachsen program of the Volkswagen Foundation (ZN4072). Computations on the HPC cluster of Osnabrück University, funded by the DFG (project 456666331).