ICLR 2024 Workshop on AI4Differential Equations

Learning Iterative Algorithms to solve PDEs

Lise Le Boudec, Emmanuel De Bézenac, Louis Serrano, Yuan Yin, Patrick Gallinari

ISIR Institut des Systèmes Intelligents et de Robotique · Sorbonne Université · CNRS · ETH zürich · CRITEO AI Lab

Motivation

➡ *Problem statement*

$$\mathcal{N}(u;\gamma) = f \quad \text{in } \Omega, \qquad \mathcal{B}(u) = g \quad \text{on } \partial\Omega.$$

Solve parametric equations with varying parameters (γ), forcing terms (f) or initial/boundary conditions (g) on a given domain Ω with boundary $\partial\Omega$.

➡ *Hypothesis*

✦ We assume access to a dataset of pairs composed of the PDE data (γ, f, g) and the associated solution u on a grid.

➡ *Relation to existing works*

✦ Physics-Informed Neural Networks [1] models the solution u as a Neural Network and uses the residual of the PDE as optimization criteria. Each PDE needs a full training of the solution.

✦ Neural Operators [2] focus on learning the solution operator directly through a single neural network pass using large amount of data.

References

[1] 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

[2] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020

Framework

➡ *Build an iterative method to solve PDEs with no data*

$$\theta^{k+1} = \mathcal{A}(\theta^k;\ \mathcal{L}_{\text{PDE}}, \gamma, f, g)$$

Ansatz

$$u_\theta(x) = \sum_{i=0}^{N} \theta_i \psi_i(x)$$

B-splines, Fourier, polynomials...

+

Physics-Informed criteria

$$\mathcal{L}_{\text{PDE}} = \mathcal{L}_{\text{Res}} + \lambda \mathcal{L}_{\text{BC}},$$

$$\mathcal{L}_{\text{Res}} = \sum_{x_j \in \Omega} |\mathcal{N}(u_\theta;\gamma)(x_j) - f(x_j)|^2,$$

$$\mathcal{L}_{\text{BC}} = \sum_{x_j \in \partial\Omega} |\mathcal{B}(u_\theta)(x_j) - g(x_j)|^2$$

+

Update Algorithm

$$\theta^{k+1} = \theta^k - \eta \mathcal{F}_\varrho(\nabla \mathcal{L}_{\text{PDE}}(\theta^k), \gamma, f, g)$$

Gradient descent, Adam...

+

➡ *Training based on data*

$$\mathcal{L}_{\text{DATA}} = \frac{1}{M} \sum_{i=1}^{M} ||u_{\theta_L} - u_i||$$

= *Hybrid solver for PDEs*

Algorithm 1: Inference

Data: $\theta^0 \in \mathbb{R}^n$, PDE (γ, f, g)
Result: $\theta^L \in \mathbb{R}^n$
for *k = 0...L-1* **do**
  $\theta^{k+1} = \theta^k - \mathcal{F}_{\varrho^*}(\nabla \mathcal{L}(\theta^k, \gamma, f, g), \gamma, f, g)$
end
return θ_L

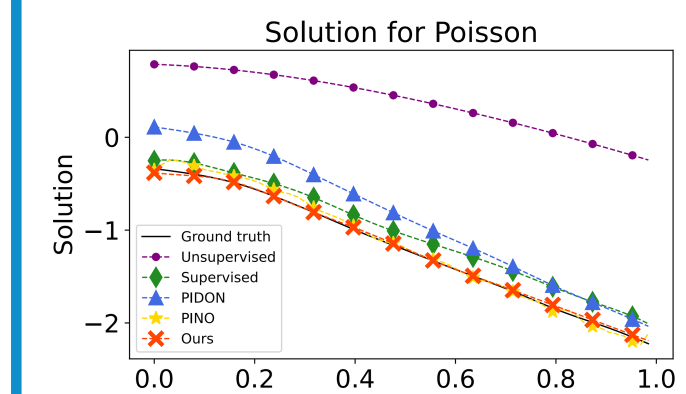
Results

➡ *Training*

	Baseline	1d Helmholtz	1d Poisson	1d + time Advection	2d Darcy-Flow
Unsupervised	*PINNs*	4.26e-1	2.33	2.26e-1	2.31e-1
Supervised	*NN*	9.04e-2	1.09e-1	1.27e-1	7.54e-3
Hybrid	*PIDON*	4.67e-1	8.84e-2	2.26e-1	4.46e-2
	PINO	4.78e-1	5.83e-3	**1.83e-4**	2.80e-2
	Ours	**1.10e-2**	**1.76e-3**	8.40e-3	**6.21e-3**

Table 1: Results - metrics in MSE on the test set.

🡅 Numerical comparison with baselines



🡄 Comparison of trajectories on Poisson

➡ *Test-time Optimization*

5 steps to compute the solution