Learning a Neural Solver for Parametric PDEs to Enhance Physics-Informed Methods

Lise Le Boudec [1], Emmanuel de Bezenac [2], Louis Serrano [1],
Ramon Daniel Regueiro-Espino [1], Yuan Yin [3], Patrick Gallinari [1,4]

[1] Sorbonne Université, CNRS, ISIR, F-75005 Paris, France

[2] INRIA Paris, France

[3] Valeo.ai, Paris, France

[4] Criteo AI Lab, Paris, France

Context

- We focus on solving **parametric** PDEs

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

- Existing methods:

Traditional solvers

+ Theoretical guaranty

- Computationally demanding

Physics-informed Neural Networks

+ No data required

- Slow to optimize
- Require one training per PDE

Data-driven Methods / Neural Operators

+ Fast inference for new PDEs

- A lot of data required

Context

- We focus on solving **parametric** PDEs

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

- Existing methods:

Traditional solvers

+ Theoretical guarantee

- Computationally demanding

Physics-informed Neural Networks

+ No data required

- Slow to optimize
- Require one training per PDE

Data-driven Methods / Neural Operators

+ Fast inference for new PDEs

- A lot of data required

Context

- We focus on solving **parametric** PDEs

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

- Existing methods:

Traditional solvers

+ Theoretical guarantee

- Computationally demanding

Physics-informed Neural Networks

+ No data required

- Slow to optimize
- Require one training per PDE

Data-driven Methods / Neural Operators

+ Fast inference for new PDEs

- A lot of data required

Context

- We focus on solving **parametric** PDEs

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

- Existing methods:

Traditional solvers

+ Theoretical guarantee

- Computationally demanding

Physics-informed Neural Networks

+ No data required

- Slow to optimize
- Require one training per PDE

Data-driven Methods / Neural Operators

+ Fast inference for new PDEs

- A lot of data required

Context

- We focus on solving **parametric** PDEs

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

- Existing methods:

Traditional solvers

+ Theoretical guarantee

- Computationally demanding

Physics-informed Neural Networks

+ No data required

- Slow to optimize
- Require one training per PDE

Data-driven Methods / Neural Operators

+ Fast inference for new PDEs

- A lot of data required

Context

- We focus on solving **parametric** PDEs

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

- Existing methods:

Physics-informed Neural Networks

+ No data required

- Slow to optimize
- Require one training per PDE

$$\mathcal{L}_{\text{PDE}} = \mathcal{L}_{\text{Res}} + \lambda \mathcal{L}_{\text{BC}}, \quad , \lambda > 0$$

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

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

Objective

Learning an iterative algorithm that efficiently solves PDEs

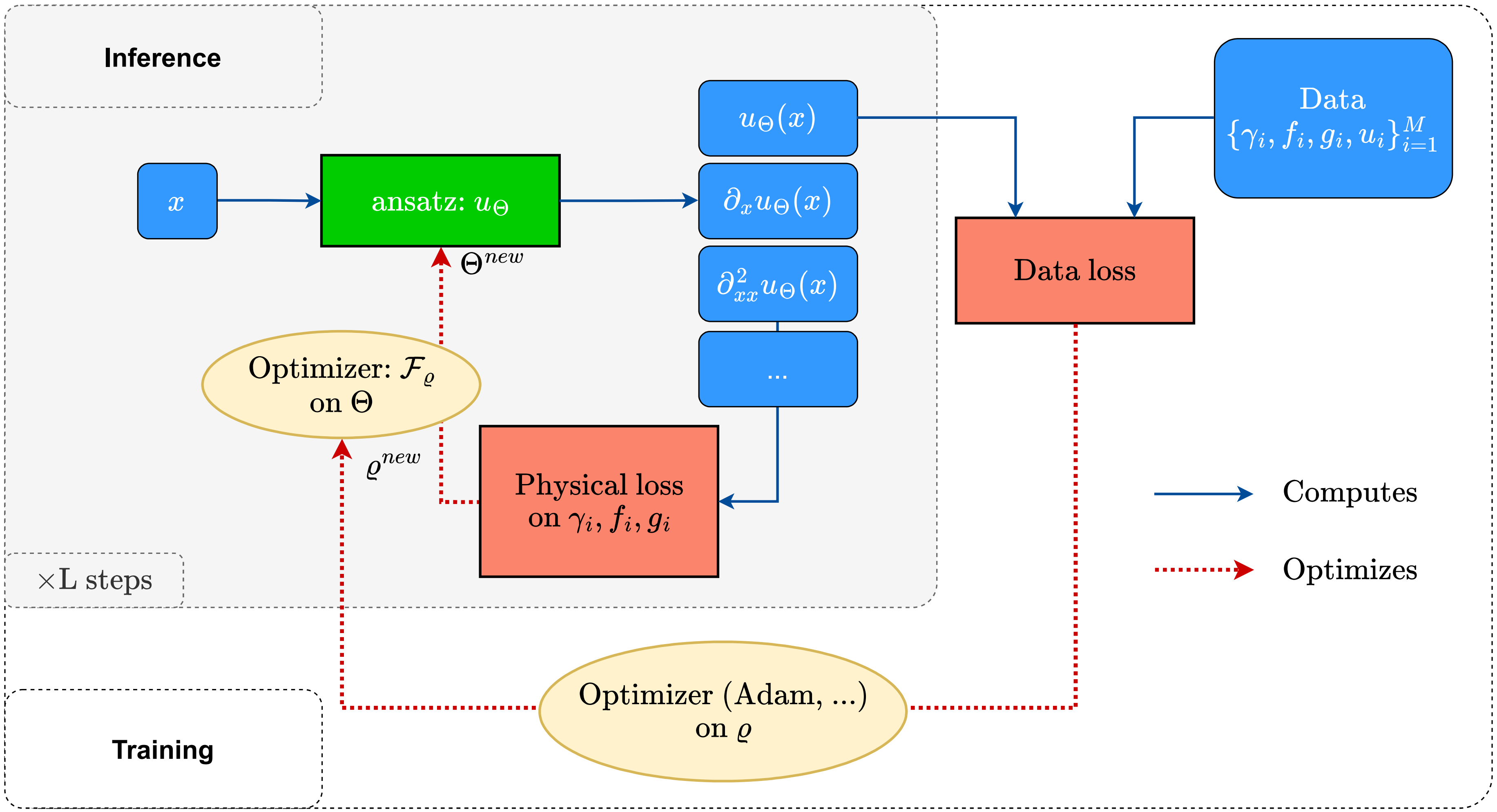
+ Solving from the PDE residual
+ Fast solving of new PDEs (L small)
+ No retraining needed

❖ Assume we have access to samples of PDE parameters and the associated target solutions, sampled at some colocation points.

Algorithm 1: Inference using the neural PDE solver.

Data: $\Theta_0 \in \mathbb{R}^n$, PDE (γ, f, g)
Result: $\Theta_L \in \mathbb{R}^n$
for l = 0...L-1 do
  $\Theta_{l+1} = \Theta_l - \eta \mathcal{F}_\varrho(\nabla \mathcal{L}_{\text{PDE}}(\Theta_l), \gamma, f, g)$
end
return Θ_L

Approach



Optimization scheme of a physics-informed method with our framework.

More about Neural solver

Code

Paper