Deep Neural Operators for Partial Differential Equations

Learning Solution Operators for Parametric PDEs

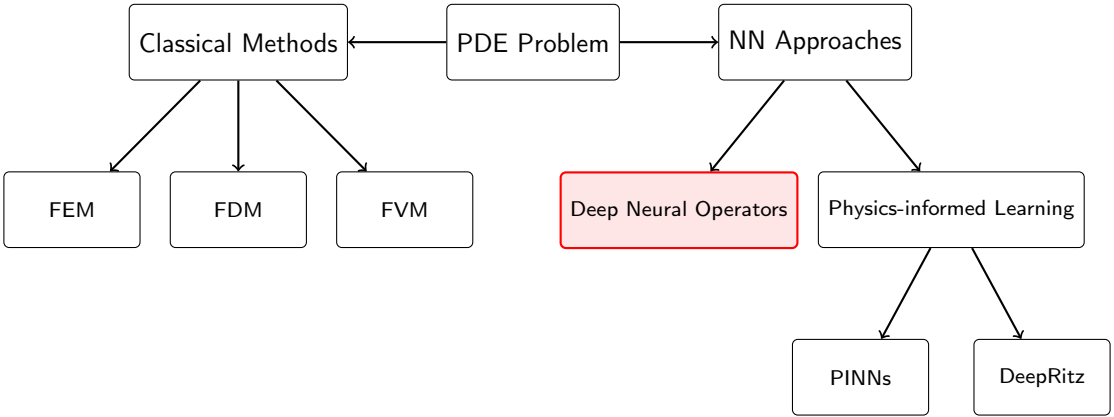
Vincent Scholz & Dr. Yaohua Zang

Professorship for Data-driven Materials Modeling

June 11, 2025

Outline

Neural Network Approaches for Solving PDEs



Example 1: The design of porous media

Figure: Fluid flows through porous media

Darcy's Flow:

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in \Omega$$

- $u(x)$: pressure field
- $a(x)$: **media with varying permeability**
- $f(x)$: source term (known)

Learning Goal:

$$\mathcal{G} : a(x) \mapsto u(x)$$

Example 2: The design of car shape

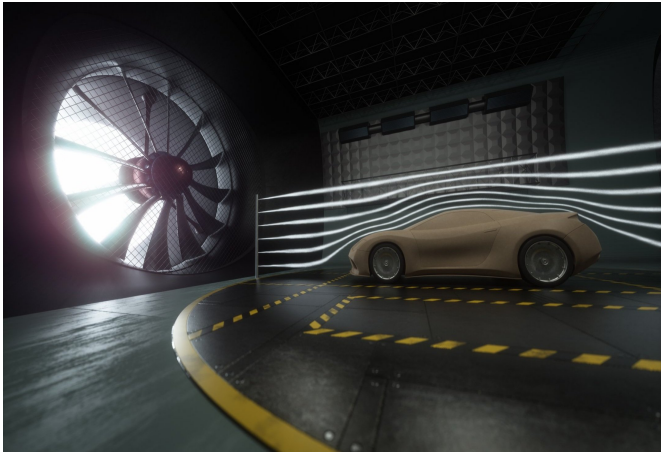


Figure: Airflow around a car

Stokes' Flow:

$$-\mu\nabla^2 \mathrm{u} + \nabla p = 0, \quad \text{in } \Omega/\Omega_{shape}$$
$$\nabla \cdot \mathrm{u} = 0, \quad \text{in } \Omega/\Omega_{shape}$$

- $u(x)$: velocity vector field
- $p(x)$: pressure field
- Ω_{shape}: **the shape of the car**

Learning Goal:

$$\mathcal{G} : \Omega_{shape} \mapsto (u(x), p(x))$$

Parametric PDEs: The General Setting

Consider a general PDE (2D) with parameter a:

$$F(x, y, u, u_x, u_y, u_{xx}, u_{yy}, \ldots, a) = 0, \quad (x, y) \in \Omega \tag{1}$$

- $u(x, y)$: **desired solution**
- $u_x, u_y, \ldots$: **partial derivatives**
- a: **parameter** (scalar, vector, or function $a(x, y)$)
- Ω: **spatial domain** with boundary conditions on $\partial\Omega$

Goal of parametric PDEs: $\mathcal{G} : a \mapsto u$

Example of parametric PDEs: Darcy's Flow with varying permeability

Darcy Flow Equation:

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in \Omega \tag{2}$$

where

- $u(x)$: pressure/temperature field
- $a(x)$: **varying permeability/conductivity**
- $f(x)$: source term (known)

Goal: *predict pressure fields $u(x)$ given different permeability/conductivity fields $a(x)$*

Applications:

- Porous material design
- Oil reservoir simulation
- Electrical impedance tomography

Example of parametric PDEs: Wave Equation with varying initial conditions

Wave Equation with Varying Initial Conditions:

$$\begin{cases} u_{tt}(x,t) - c^2 \Delta u(x,t) = 0, & x \in [0,1]^2, \ t \in [0,T] \\ u(x,0) = a(x), & \text{initial condition} \\ u_t(x,0) = 0 & \text{zero initial velocity} \end{cases} \tag{3}$$

where

- $u(x,t)$: wave displacement
- $a(x)$: **initial wave** (parameter)
- $c(x)$: **wave speed** (constant or parameter)

Goal: *predict the final wave fields $u(x,T)$ given different initial wave $a(x)$ or wave speeds $c(x)$*

Applications: Seismology, acoustics, signal propagation

Function-to-Function Mapping

For a parametric PDE system:

$$F(x, u, \nabla u, \Delta u, \ldots, \mathbf{a}) = 0, \quad x \in \Omega$$

A Deep Neural Operator learns:

$$\mathcal{G}_\theta(a)(x) : a(x) \in \mathcal{A} \longrightarrow u(x) \in \mathcal{U} \tag{4}$$

where

- $\mathcal{A}$: **Banach space** of input functions
- $\mathcal{U}$: **Banach space** of output functions
- $\mathcal{G}_\theta$: **Parametrized neural operator** with parameters θ

Operator Mapping : Input function $a(x) \rightarrow$ Output function $u(x)$

Deep Neural Operators vs. Physics-informed Learning

Neural Operator Learning:

- Learn the **solution operator** of parametric PDEs
- Handle **entire families** of PDEs (different parameters)
- Fast inference for new parameters

Physics-informed Learning:

- Solve **one PDE instance** at a time
- Each new parameter a requires full numerical solution
- Computationally expensive for many-query problems

Goal of Neural Operator Learning:

$$\mathcal{G} : a(x) \to u(x)$$

Goal of Physics-informed Learning:

$$u : x \to u(x)$$

Deep Neural Operators for Parametric PDEs: Supervised Learning

Supervised Learning Framework:

Training Data: Paired function observations (a, u)

- $a(x)$: input function (permeability, initial condition, etc.)
- $u(x)$: corresponding PDE solution

Data Sources:

- **High-precision numerical solvers** (e.g., finite element methods, finite difference methods)
- **Experimental measurements**

Goal: Learn $\mathcal{G}_\theta$ that generalizes to **unseen functions** $a(x)$

Key Advantages of Deep Neural Operators (DNOs)

By working in **infinite-dimensional functional space**, DNOs provide:

1. **Resolution Independence**
   - Generalize across arbitrary spatial resolutions
   - Train on coarse mesh, predict on fine mesh
2. **Geometric Flexibility**
   - Handle varying geometries and domains
   - Adapt to different boundary conditions
3. **Fast Inference**
   - Real-time solution prediction
   - Ideal for many-query problems

Perfect for: Uncertainty quantification, Bayesian inverse problems, real-time applications

Comparison: DNOs vs Solution Operator Networks

Aspect	Solution Operator Network	Deep Neural Operator
Input	Discretized vector $\vec{a} \in \mathbb{R}^m$	Function $a(x) \in \mathcal{A}$
Output	Discretized vector $\vec{u} \in \mathbb{R}^n$	Function $u(x) \in \mathcal{U}$
Mapping	Vector-to-vector	**Function-to-function**
Training Data	Paired vectors $(\vec{a}, \vec{u})$	Paired functions $(a(x), u(x))$
Flexibility	Limited to specific discretizations	**Mesh-independent**

Solution Operator: $\mathcal{K}_\theta(\vec{a}) : \vec{a} \in \mathbb{R}^m \to \vec{u} \in \mathbb{R}^n$

DNO: $\mathcal{G}_\theta(a) : a(x) \in \mathcal{A} \to u(x) \in \mathcal{U}$

Key Differences: DNOs vs Solution Operator Networks

Solution Operator Networks:

- Fixed input/output dimensions
- Tied to specific mesh resolution
- Cannot generalize across discretizations
- Limited geometric flexibility

Discretization-dependent

DNO Approach:

- Works with continuous functions
- Resolution independent
- Generalizes across meshes
- Handles varying geometries

Maintains the continuous nature of PDEs

DNOs: Greater flexibility + Better generalization

Summary: Why Deep Neural Operators?

Paradigm Shift: From solving individual PDEs to learning solution operators

Key Benefits:

1. **Efficiency:** Train once, infer many times
2. **Flexibility:** Resolution and geometry independent
3. **Generalization:** Handle entire families of PDEs
4. **Speed:** Real-time inference for new parameters

Ideal Applications:

- Many-query problems (optimization, uncertainty quantification)
- Real-time systems
- Parametric studies
- Inverse problems

What's Next?

Upcoming Topics:

1. **Fourier Neural Operators (FNOs)**
   - Architecture details
   - Fourier transforms in neural networks
2. **DeepONets**
   - Branch and trunk networks
   - Universal approximation properties
3. **Implementation and Training**
   - Practical considerations
   - Code examples

Questions?

Deep Neural Operators for PDEs

Thank you for your attention!