Introduction to (Data-driven) Deep Neural Operators

Yaohua Zang

June 23, 2025

0.1 Why Should We Consider Neural Operators? – Motivation Behind Deep Neural Operators

In this section, we explore the motivation for **Deep Neural Operators (DNOs)** and the class of problems they aim to solve. Unlike traditional deep learning methods that are trained to solve a single PDE instance, neural operators are trained to learn the **solution operator** for **an entire family of PDEs** with varying inputs or parameters.

0.1.1 Parametric PDEs and the Need for Operator Learning

Many real-world physical systems are governed by parameterized PDEs, where certain coefficients, source terms, or boundary conditions vary across different scenarios. Consider the general form of a PDE:

$$F(x, y, u, u_x, u_y, u_{xx}, u_{yy}, \cdots, \mu) = 0, \quad (x, y) \in \Omega$$

- Here, $u(x, y)$ is the desired solution.
- $u_x, u_y, \cdots$ are partial derivatives of u.
- μ represents the parameter of the PDE, which can be a scalar, a vector, or a function $\mu(x, y)$.
- The PDE is defined on a spatial domain Ω, with appropriate boundary conditions prescribed on $\partial\Omega$ (Dirichlet, Neumann, or mixed).

Problem Setting: Parametric PDEs The task is to find the solution $u(x, y; \mu)$ for any given parameter μ. This naturally defines a mapping (operator) from the parameter space to the solution space:

$$\mu(x, y) \mapsto u(x, y; \mu)$$

This is referred to as the **solution operator**, which we denote as:

$$\mathcal{G} : \mu \mapsto u$$

Learning this operator is the central goal of **Deep Neural Operators (DNO)**.

0.1.2 Real-World Examples of Parametric PDE Problems

To better understand this setup, let's consider two specific PDE models where learning such an operator would be useful.

Example 1: Darcy Flow with Varying Permeability The Darcy flow equation models fluid movement through porous media and takes the form:

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

- $u(x)$: pressure or temperature field (solution)
- $a(x)$: spatially varying permeability or conductivity
- $f(x)$: source term (known)

Objective: Given different instances of $a(x)$, find the corresponding solution $u(x)$.
This defines a solution operator:

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

Such problems frequently arise in hydrology, oil reservoir simulation, and electrical impedance tomography (EIT).

Example 2: Wave Equation with Different Initial Conditions Consider the wave equation:

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

- $u(x,t)$: wave displacement at position x and time t
- $a(x)$: initial input (parameter)
- c: constant wave speed

Objective: Predict the wave field at the final time T, i.e., compute:

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

This problem is relevant to seismology, acoustics, and signal propagation in materials.

0.2 What Are Deep Neural Operators?

Now that we've discussed the motivation and context for operator learning, let's formally define what Deep Neural Operators (DNOs) are and how they differ from traditional approaches.

0.2.1 What Does a DNO Learn?

A Deep Neural Operator is designed to learn a mapping between function spaces — not just fixed vectors or pointwise solutions. Specifically, it learns an operator that maps input functions to output functions:

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

- $\mathcal{A}$: Banach space of input functions (e.g., coefficient functions $a(x)$)
- $\mathcal{U}$: Banach space of output functions (e.g., PDE solutions $u(x)$)

- $\mathcal{G}_\theta$: A **parametrized neural operator** with learnable parameters θ

This is a **function-to-function** mapping:

- The **input** is a function $a(x)$
- The **output** is also a function $u(x)$

0.2.2 Supervised Learning with Function Pairs

Training a DNO requires many **paired observations** of the form (a, u), where:

- $a(x)$: input function (e.g., permeability, initial condition)
- $u(x)$: output solution from the PDE

These pairs are typically obtained from:

- **High-precision Numerical solvers** (e.g., finite element methods)
- **Real experimental measurements**

As such, the training of DNOs falls under supervised learning, where the goal is to learn a model $\mathcal{G}_\theta$ that generalizes well to unseen functions $a(x)$.

0.2.3 The Advantage of DNO

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

- Generalize across arbitrary spatial resolutions
- Handle varying geometries or domains
- Offer fast inference for unseen PDE instances

This makes them especially powerful in scenarios requiring **real-time solution prediction**, **uncertainty quantification**, or **many-query problems** like Bayesian inverse problems.

0.3 DNOs vs Traditional Solution Operator Networks

To better understand DNOs, let's compare them with a more traditional deep learning method used for solving parametric PDEs: the **solution operator network**.

	Solution Operator Network	**Deep Neural Operator (DNO)**
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 Type	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, potentially continuous resolution

The **solution operator network** learns a mapping:

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

This model is discretization-dependent: the input and output are both vectors of fixed size, tied to a mesh resolution. In contrast, DNOs learn directly in function space and are not limited by a fixed discretization, offering **greater flexibility** and **potential for generalization** to different resolutions.