# Physics-Informed Neural Networks and Neural Operators for Parametric PDEs: Methods, Applications and Future Directions

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

PDEs arise ubiquitously in science and engineering, where solutions depend on parameters representing physical properties, boundary conditions, or geometric configurations. Traditional numerical methods require solving the PDE anew for each parameter value, making parameter space exploration prohibitively expensive for high-dimensional problems. Recent advances in machine learning, particularly physics-informed neural networks (PINNs) and neural operators, have revolutionized parametric PDE solving by learning solution operators that generalize across parameter spaces. We critically analyze two main paradigms: (1) PINNs, which embed physical laws as soft constraints and excel at inverse problems with sparse data, and (2) neural operators (including DeepONet, Fourier Neural Operator, and their variants), which learn mappings between infinite-dimensional function spaces and achieve unprecedented parameter space generalization. Through detailed comparisons across fluid dynamics, solid mechanics, heat transfer, and electromagnetics, we show that neural operators can achieve computational speedups ranging from $10^3$ to $10^5$ times faster than traditional solvers for multi-query scenarios, while maintaining comparable accuracy. We provide practical guidance for method selection, discuss theoretical foundations including universal approximation and convergence guarantees, and identify critical open challenges including high-dimensional parameter spaces, complex geometries, and out-of-distribution generalization. This work establishes a unified framework for understanding parametric PDE solvers through the lens of operator learning, offering a comprehensive resource—which we intend to incrementally update—for this rapidly evolving field.

**Keywords:** Parametric PDEs, Physics-Informed Neural Networks, Neural Operators, Scientific Machine Learning, Operator Learning

## NOMENCLATURE

| Symbol | Meaning |
|---|---|
| $\mu$ | Parameter vector |
| $\mathcal{P}$ | Parameter space |
| $d$ | Parameter dimension |
| $\Omega$ | Spatial domain |
| $u(x, t; \mu)$ | Parameter-dependent PDE solution |
| $\mathcal{L}(\cdot; \mu)$ | Parameter-dependent differential operator |
| $\mathcal{G}$ | Solution operator mapping parameters to solutions |
| $\mathcal{A}$ | Input function space |
| $\mathcal{U}$ | Solution function space |
| $\theta$ | Neural network parameters |
| $\mathcal{L}_{\text{PDE}}$ | Physics-informed loss |
| $\mathcal{L}_{\text{data}}$ | Data fidelity loss |
| Re | Reynolds number |
| DOF | Degrees of freedom |

# 1 Introduction and Background

## 1.1 Motivation: The Parametric Challenge in Scientific Computing

Parametric partial differential equations (PDEs) represent one of the most fundamental yet computationally challenging problems in scientific computing and engineering. These equations, whose solutions depend on parameters representing physical properties, boundary conditions, or geometric configurations, arise ubiquitously across disciplines: from material design requiring exploration of compositional parameter spaces, to fluid dynamics demanding Reynolds number sweeps, to uncertainty quantification necessitating sampling over parameter distributions.

Consider a canonical example from computational fluid dynamics: the flow around an airfoil depends critically on parameters such as the Reynolds number, angle of attack, and airfoil shape. Traditional computational fluid dynamics (CFD) solvers, while mature and reliable, must solve the governing Navier-Stokes equations independently for each parameter configuration. For a modest parameter space exploration involving just three parameters with 10 values each, this requires 1,000 independent simulations—a prohibitively expensive endeavor when each simulation demands hours or days of computation on high-performance computing clusters.

This "one-parameter-at-a-time" limitation of traditional numerical methods becomes even more severe in several critical scenarios:

**Real-time decision making:** In applications such as autonomous systems, medical diagnostics, or process control, solutions must be obtained in milliseconds rather than hours. For instance, patient-specific cardiovascular simulations for surgical planning cannot wait for overnight cluster computations.

**Inverse problems and parameter identification:** Determining unknown physical parameters from observational data—such as inferring material properties from experimental measurements or identifying hidden forcing terms in climate models—requires repeatedly solving forward problems with different parameter guesses. Traditional optimization loops involving thousands of PDE solves become computationally intractable.

**Uncertainty quantification (UQ):** Modern engineering design must account for uncertainties in material properties, manufacturing tolerances, and operational conditions. Monte Carlo methods for UQ typically require $10^4$-$10^6$ PDE evaluations to accurately estimate probability distributions—a task infeasible with conventional solvers for complex systems.

**High-dimensional parameter spaces:** Many real-world problems involve tens or hundreds of parameters. For example, weather forecasting models contain hundreds of thousands of uncertain parameters, molecular dynamics simulations depend on high-dimensional potential energy surfaces, and topology optimization problems search over spaces of possible geometric configurations. Traditional methods suffer from the curse of dimensionality, where sampling complexity grows exponentially with parameter dimension.

The scientific and economic implications are profound. The aviation industry spends billions on wind tunnel testing and CFD simulations for aircraft design optimization. The pharmaceutical sector requires years of computational protein folding studies for drug discovery. Climate modeling centers consume enormous computational resources for ensemble forecasts that still struggle to quantify uncertainties adequately.

**The Central Thesis:** While parametric PDEs are ubiquitous in science and engineering, traditional numerical methods solve one parameter configuration at a time, making parameter space exploration prohibitively expensive. This survey examines how machine learning methods—particularly physics-informed neural networks and neural operators—are transforming our ability to solve parametric PDEs by learning solution operators that generalize across entire parameter spaces. For multi-query scenarios where solutions are needed at many parameter values, these methods enable computational speedups of $10^3$ to $10^5$ times compared to traditional solvers, with documented examples including 60,000$\times$ speedup for turbulence (Li et al., 2021) and 1,000$\times$ speedup for laminar flows (Jin et al., 2021).

## 1.2 Mathematical Formulation of Parametric PDEs

To establish precise terminology and mathematical foundations, we formally define the class of parametric PDEs that forms the focus of this survey.

**Definition 1** (Parametric PDE). *Let $\mathcal{P} \subset \mathbb{R}^d$ denote a parameter space of dimension $d$. A parametric PDE is defined as:*

$$\mathcal{L}(u(x,t;\mu);\mu) = f(x,t;\mu), \quad \forall \mu \in \mathcal{P}, \quad (x,t) \in \Omega \times [0,T], \tag{1}$$

*equipped with parameter-dependent boundary and initial conditions:*

$$\mathcal{B}(u(x,t;\mu);\mu) = g(x,t;\mu), \quad (x,t) \in \partial\Omega \times [0,T], \tag{2}$$

$$u(x,0;\mu) = u_0(x;\mu), \quad x \in \Omega, \tag{3}$$

*where $u : \Omega \times [0,T] \times \mathcal{P} \to \mathbb{R}^n$ is the parameter-dependent solution (possibly vector-valued), $\mathcal{L}(\cdot;\mu)$ is a parameter-dependent differential operator, $\mathcal{B}(\cdot;\mu)$ specifies boundary conditions, and $f, g, u_0$ represent parameter-dependent forcing, boundary, and initial data.*

This formulation encompasses a vast range of problems. The parameter $\mu$ can represent:

**1. Physical parameters:** Material properties (viscosity, thermal conductivity, elastic moduli), flow regime indicators (Reynolds, Rayleigh, Péclet numbers), or forcing magnitudes.

*Example:* The parameterized Burgers equation models nonlinear wave propagation with viscosity parameter $\nu$:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad \nu \in [\nu_{\min}, \nu_{\max}] \subset \mathbb{R}_+ \tag{4}$$

Here $\mathcal{P} = [\nu_{\min}, \nu_{\max}]$ represents the one-dimensional parameter space of viscosities.

**2. Geometric parameters:** Domain shapes, boundary configurations, or obstacle positions.

*Example:* Flow around parameterized airfoils, where $\mu$ represents coefficients in a geometric parameterization (e.g., NACA profile parameters or Bézier control points):

$$\begin{cases} -\nabla \cdot (\nu \nabla u) + (u \cdot \nabla)u + \nabla p = 0 & \text{in } \Omega(\mu) \\ \nabla \cdot u = 0 & \text{in } \Omega(\mu) \\ u = u_\infty & \text{on } \partial\Omega_{\text{inlet}} \\ u = 0 & \text{on } \partial\Omega_{\text{airfoil}}(\mu) \end{cases} \tag{5}$$

where the domain $\Omega(\mu)$ and airfoil boundary $\partial\Omega_{\text{airfoil}}(\mu)$ depend explicitly on shape parameters.

**3. Boundary and initial condition parameters:** Inlet velocities, temperature distributions, loading patterns, or initial state configurations.

*Example:* Heat conduction with parameterized boundary temperature:

$$\begin{cases} \rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) & \text{in } \Omega \\ T = T_{\text{wall}}(\mu) & \text{on } \partial\Omega \end{cases} \tag{6}$$

where $\mu$ parameterizes the boundary temperature profile.

**4. Source term parameters:** External forcing magnitudes, distributions, or time-dependent profiles.

The **solution manifold** is a central concept:

$$\mathcal{M} = \{u(\cdot,\cdot;\mu) : \mu \in \mathcal{P}\} \subset \mathcal{U} \tag{7}$$

where $\mathcal{U}$ is an appropriate function space (e.g., Sobolev space). The manifold $\mathcal{M}$ represents the set of all possible solutions as parameters vary. Understanding the geometric and topological properties of $\mathcal{M}$ is crucial for developing efficient approximation methods.

From an operator perspective, we seek to approximate the **parameter-to-solution map** (or solution operator):

$$\mathcal{G} : \mathcal{P} \to \mathcal{U}, \quad \mu \mapsto u(\cdot,\cdot;\mu) \tag{8}$$

This operator encodes the complete input-output relationship of the parametric PDE. Traditional numerical methods construct $\mathcal{G}$ implicitly by solving equation equation 1 for each query $\mu$. In contrast, machine learning approaches aim to learn an explicit approximation $\mathcal{G}_\theta \approx \mathcal{G}$ that can be rapidly evaluated.

**Key Challenges in Parametric PDEs:**

*High-dimensional parameter spaces:* When $d \gg 1$, the curse of dimensionality emerges. Uniform sampling of a $d$-dimensional space with $n$ points per dimension requires $n^d$ samples—exponentially expensive. Even advanced techniques like sparse grids or reduced basis methods face limitations beyond $d \sim 20$.

*Nonlinear parameter dependence:* The solution $u(\cdot, \cdot; \mu)$ may depend nonlinearly, non-smoothly, or even discontinuously on $\mu$. For instance, solutions may exhibit bifurcations, phase transitions, or shock formations at critical parameter values, making smooth interpolation impossible.

*Multi-scale phenomena:* Parameters often control multiple length or time scales simultaneously. A small change in Reynolds number can trigger transition from laminar to turbulent flow, introducing fine-scale structures that traditional coarse approximations cannot capture.

*Geometry-parameter coupling:* When geometry itself is parameterized, standard grid-based methods struggle. Each parameter value may require mesh regeneration, and solutions on different meshes cannot be directly compared or interpolated.

The mathematical challenge can be formally stated: *Given finite computational resources and training data* $\{(\mu_i, u_i)\}_{i=1}^N$ *where* $u_i = u(\cdot, \cdot; \mu_i)$*, construct an approximation* $\mathcal{G}_\theta$ *such that* $\|\mathcal{G}(\mu) - \mathcal{G}_\theta(\mu)\|_{\mathcal{U}} < \epsilon$ *for all* $\mu \in \mathcal{P}$*, with rapid evaluation* $\mathcal{O}(1)$ *ms per query.*

### 1.3 Traditional Approaches: Reduced Order Models

Before examining modern machine learning methods, we review traditional approaches to parametric PDEs, which provide both historical context and performance baselines. Classical reduced order models (ROMs) have been developed over decades, establishing rigorous mathematical foundations that inform contemporary developments.

#### 1.3.1 Proper Orthogonal Decomposition and Galerkin Projection

The Proper Orthogonal Decomposition (POD), also known as Principal Component Analysis or Karhunen-Loève expansion, forms the cornerstone of traditional ROM approaches (Holmes et al., 1996; Berkooz et al., 1993). Given snapshot solutions $\{u(\cdot, \cdot; \mu_i)\}_{i=1}^N$ collected at training parameters, POD constructs an optimal low-dimensional approximation subspace.

The method computes spatial modes $\{\phi_k\}_{k=1}^{N_{\text{POD}}}$ by solving the eigenvalue problem:

$$C\phi_k = \lambda_k \phi_k, \quad C_{ij} = \langle u_i, u_j \rangle_{\mathcal{U}} \tag{9}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{U}}$ denotes an appropriate inner product. Solutions for new parameters are approximated as:

$$u(x, t; \mu) \approx \sum_{k=1}^{N_{\text{POD}}} a_k(t; \mu) \phi_k(x) \tag{10}$$

where coefficients $a_k(t; \mu)$ are determined by Galerkin projection of the governing PDE onto the reduced space.

**Strengths:** POD-Galerkin excels when solutions lie near a low-dimensional linear subspace, achieving exponential convergence for smooth parameter dependence. Online evaluation is extremely fast ($\mathcal{O}(\text{ms})$) once modes are computed.

**Limitations for Parametric Problems:** *Linear assumption:* POD constructs a linear subspace, failing when the solution manifold $\mathcal{M}$ is intrinsically nonlinear (as in transport-dominated or bifurcating systems)., *Parameter-dependent training:* Modes depend on the specific parameter samples chosen. Poor parameter space coverage leads to inadequate approximations., and *Dimensionality limits:* While effective for $d < 10$, performance degrades rapidly in higher dimensions without adaptive sampling.

#### 1.3.2 Reduced Basis Methods

Reduced basis (RB) methods (Quarteroni et al., 2015; Hesthaven et al., 2016; Rozza et al., 2008) extend POD through sophisticated greedy algorithms for parameter space exploration. The key innovation is adaptive selection of snapshot parameters to maximize approximation quality across $\mathcal{P}$.

The greedy algorithm iteratively identifies the parameter $\mu^*$ where the current reduced model has largest error:

$$\mu^{n+1} = \arg\max_{\mu \in \mathcal{P}} \eta(\mu; \{\phi_k\}_{k=1}^n) \tag{11}$$

where $\eta(\mu; \cdot)$ is a computable error estimator. A new solution snapshot at $\mu^{n+1}$ is computed and orthogonalized against existing basis functions.

**Strengths:** Rigorous a posteriori error bounds, systematic parameter space coverage, and theoretical optimality guarantees under affine parameter dependence.

**Limitations:** Despite these advantages, RB methods face several constraints. First, they require affine parameter dependence, meaning PDE operators must decompose as $\mathcal{L}(\cdot;\mu) = \sum_{q=1}^{Q} \Theta_q(\mu)\mathcal{L}_q(\cdot)$ with parameter-independent operators $\mathcal{L}_q$. This requirement excludes geometry-parameterized problems and many nonlinear cases. Second, the greedy procedure incurs substantial offline computational expense, requiring dozens to hundreds of full-order model solves. Third, like POD, RB methods face challenges beyond $d \sim 20$ despite sophisticated sampling strategies.

### 1.3.3 Sparse Grid and Multi-Level Methods

For higher-dimensional parameter spaces, sparse grid methods based on Smolyak's algorithm (Bungartz & Griebel, 2004) provide a middle ground between full tensor-product grids and random sampling. The key idea is anisotropic refinement, placing more collocation points along directions of high solution variability.

Multi-level Monte Carlo (MLMC) (Giles, 2015) tackles high-dimensional uncertainty quantification by combining low-fidelity (coarse mesh) solutions with high-fidelity corrections, achieving variance reduction without prohibitive cost.

**Limitations:** Sparse grids require smoothness assumptions and struggle beyond $d \sim 30$. MLMC helps with UQ but does not provide a general-purpose surrogate for $\mathcal{G}$.

### 1.3.4 Comparative Performance Analysis

Table 1 summarizes the capabilities and limitations of traditional ROM approaches.

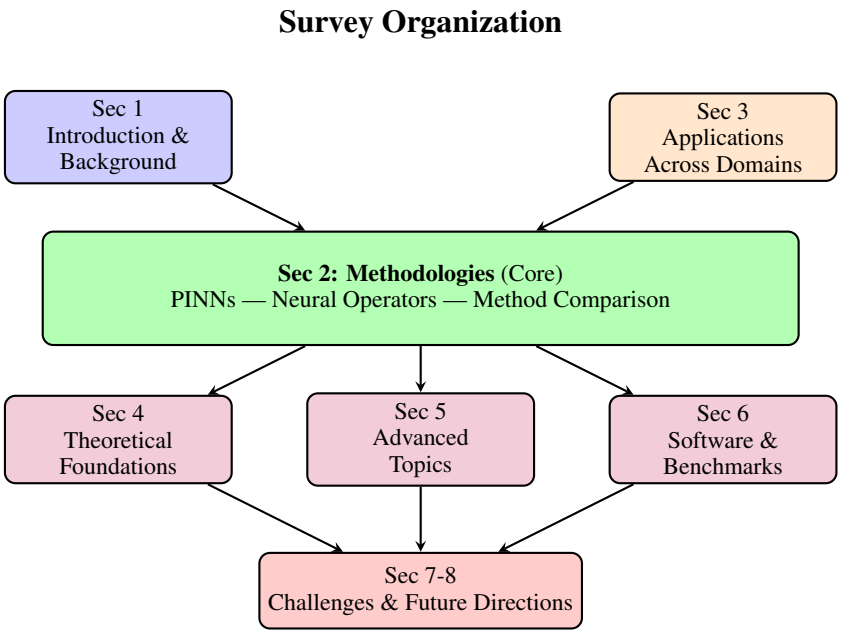


Figure 1: Survey organization and reading roadmap.

Table 1: Comparison of traditional reduced order modeling approaches for parametric PDEs

| Method | Parameter Dimension | Online Query Speed | Offline Training Cost | Main Limitations |
|---|---|---|---|---|
| POD-Galerkin | Low ($d < 10$) | Fast ($\sim$ms) | Moderate (snapshot collection) | Linear subspace assumption; parameter coverage dependency |
| Reduced Basis | Low ($d < 10$) | Very Fast ($\sim$ms) | High (greedy iterations) | Affine parameter dependence; geometry limitations |
| Sparse Grids | Medium ($d < 20 - 30$) | Moderate | High (collocation points) | Smoothness requirements; curse of dimensionality persists |
| Multi-Level MC | High ($d > 50$) | N/A (statistical) | Very High | UQ-specific; no surrogate model |

**The Gap and Opportunity:** Traditional ROMs have achieved remarkable success in low-dimensional, smooth parameter regimes with structured problems. However, they face fundamental barriers for (1) high-dimensional parameter spaces ($d > 50$), (2) geometry-parameterized problems, (3) nonlinear/non-smooth parameter dependence, and (4) multi-scale phenomena requiring resolution adaptivity.

Machine learning approaches, particularly physics-informed neural networks and neural operators, promise to overcome these limitations through flexible nonlinear representations, mesh-free formulations, and data-driven adaptivity. The transition from linear subspace methods to nonlinear operator learning represents a paradigm shift in computational science.

Having established the mathematical foundations and traditional baseline methods, we now turn to the main focus of this survey: how modern machine learning approaches revolutionize parametric PDE solving. The survey organization is shown in Figure 1.

## 2 Methodologies: Physics-Informed Neural Networks and Neural Operators

This section forms the technical core of the survey, systematically examining methods organized by their fundamental approach to parametric PDE solving. We distinguish two main paradigms: (1) *physics-informed neural networks* (PINNs), which embed governing equations as soft constraints and typically require retraining for each parameter region, and (2) *neural operators*, which learn mappings between infinite-dimensional function spaces and naturally generalize across parameter spaces through a single training phase. This survey covers traditional and neural methods (see Figure 2).

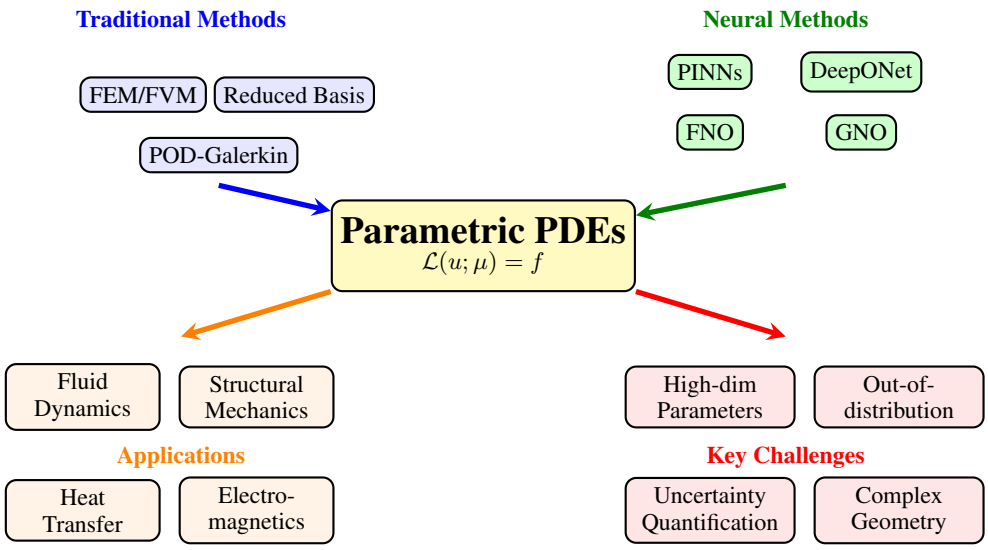


Figure 2: Research landscape of parametric PDE solving methods.

### 2.1 Physics-Informed Neural Networks (PINNs)

#### 2.1.1 Foundational Framework

Physics-informed neural networks, pioneered by Raissi et al. (Raissi et al., 2019) in their seminal 2019 work, represent a breakthrough in integrating physical laws with data-driven learning. The core innovation lies in encoding PDE residuals directly into the loss function, enabling networks to discover solutions satisfying both data constraints and governing equations.

For a parametric PDE of the form equation 1, a PINN approximates the solution as:

$$u_\theta(x, t, \mu) \approx u(x, t; \mu) \tag{12}$$

where $\theta$ denotes the neural network parameters (weights and biases). The network is trained by minimizing a composite loss function:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}} \tag{13}$$

where each term enforces different physical and data constraints:

**Physics Residual Loss:** Measures PDE satisfaction at collocation points:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} \left| \mathcal{L}(u_\theta(x_i, t_i, \mu_i); \mu_i) - f(x_i, t_i; \mu_i) \right|^2 \tag{14}$$

where $\{(x_i, t_i, \mu_i)\}$ are randomly sampled collocation points in the spatiotemporal-parameter domain.

**Boundary Condition Loss:** Enforces boundary constraints:

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{j=1}^{N_{\text{BC}}} \left| \mathcal{B}(u_\theta(x_j, t_j, \mu_j); \mu_j) - g(x_j, t_j; \mu_j) \right|^2 \tag{15}$$

**Initial Condition Loss:** For time-dependent problems:

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{k=1}^{N_{\text{IC}}} \left| u_\theta(x_k, 0, \mu_k) - u_0(x_k; \mu_k) \right|^2 \tag{16}$$

**Data Fidelity Loss:** Matches available observations:

$$\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{\ell=1}^{N_{\text{data}}} \left| u_\theta(x_\ell, t_\ell, \mu_\ell) - u_\ell^{\text{obs}} \right|^2 \tag{17}$$

The automatic differentiation capability of modern deep learning frameworks enables efficient computation of PDE residuals through repeated backpropagation.

**Parametric Extension Strategies:**

For parametric problems, three main approaches incorporate parameters into the PINN architecture:

*Strategy 1: Parameter as Additional Input* (Most Common)

$$u_\theta(x, t, \mu) \leftarrow \text{NN}_\theta(\text{concat}([x, t, \mu])) \tag{18}$$

Parameters are concatenated with spatiotemporal coordinates as network inputs. This direct approach allows a single network to handle multiple parameter values simultaneously.

*Strategy 2: Multi-Task Learning Framework*

$$u_\theta(x, t, \mu) = h_{\theta_{\text{shared}}}(x, t) + \sum_{k=1}^{K} \alpha_k(\mu; \theta_{\text{task}}) \psi_k(x, t; \theta_{\text{task}}) \tag{19}$$

A shared feature extractor learns common structures, while parameter-specific heads capture unique behaviors. This architecture exploits commonalities across parameter values.

*Strategy 3: Conditional Neural Networks*

$$u_\theta(x, t, \mu) = \text{NN}_{\theta(\mu)}(x, t) \tag{20}$$

where $\theta(\mu) = g_\phi(\mu)$ is a hypernetwork that generates weights conditioned on $\mu$. This approach provides maximum flexibility but increases training complexity.

**Example: Parametric Burgers Equation**

To illustrate, consider the viscous Burgers equation:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad x \in [0, 2\pi], \quad t \in [0, 1] \tag{21}$$

**Algorithm 1** Parametric PINN Training

**Input:** PDE operator $\mathcal{L}$, parameter space $\mathcal{P}$, domain $\Omega \times [0, T]$
**Initialize:** Neural network $u_\theta$ with random weights
**for** epoch $= 1$ to $N_{\text{epochs}}$ **do**
  Sample collocation points: $\{(x_i, t_i, \mu_i)\}_{i=1}^{N_{\text{PDE}}}$
  Sample boundary points: $\{(x_j, t_j, \mu_j)\}_{j=1}^{N_{\text{BC}}}$
  Sample initial points: $\{(x_k, 0, \mu_k)\}_{k=1}^{N_{\text{IC}}}$
  **for** each point $(x, t, \mu)$ **do**
    Compute $u = u_\theta(x, t, \mu)$ via forward pass
    Compute $\frac{\partial u}{\partial t}, \nabla u, \nabla^2 u$ via automatic differentiation
    Evaluate residual: $r = \mathcal{L}(u; \mu) - f(x, t; \mu)$
  **end for**
  Compute total loss: $\mathcal{L}_{\text{total}}$ via Eq. equation 13
  Update $\theta$ via gradient descent: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{total}}$
**end for**
**Return:** Trained network $u_\theta$

with viscosity parameter $\nu \in [0.01, 0.1]$ and initial condition $u(x, 0) = -\sin(x)$.

The PINN network takes inputs $(x, t, \nu) \in [0, 2\pi] \times [0, 1] \times [0.01, 0.1]$ and outputs $u_\theta(x, t, \nu)$. The physics loss becomes:

$$\mathcal{L}_{\text{PDE}} = \mathbb{E}\left[\left(\frac{\partial u_\theta}{\partial t} + u_\theta \frac{\partial u_\theta}{\partial x} - \nu \frac{\partial^2 u_\theta}{\partial x^2}\right)^2\right] \tag{22}$$

where derivatives are computed via automatic differentiation and expectation is approximated via Monte Carlo sampling over $(x, t, \nu)$.

### 2.1.2 Parametric Enhancement Techniques

Building on the foundational framework, researchers have developed numerous techniques to improve PINN performance for parametric problems. We examine the most impactful developments:

**1. Inverse Problems and Parameter Identification**

The pioneering work of Raissi et al. (Raissi et al., 2019), building on foundational deep learning approaches for PDEs (Han et al., 2018; Sirignano & Spiliopoulos, 2018; Berg & Nyström, 2018; E & Yu, 2018), demonstrated PINNs' remarkable capability for inverse problems: inferring unknown parameters from sparse observational data. Complementary methodological developments (Raissi, 2018; Cuomo et al., 2022; Lu et al., 2022) have further established PINNs as powerful tools for parameter identification. This capability is particularly valuable when parameters represent hidden physical properties.

*Framework:* Parameters $\mu$ become learnable quantities alongside network weights $\theta$. The modified loss function:

$$\mathcal{L}_{\text{total}}(\theta, \mu) = \mathcal{L}_{\text{PDE}}(\theta, \mu) + \mathcal{L}_{\text{data}}(\theta, \mu) + \mathcal{R}(\mu) \tag{23}$$

where $\mathcal{R}(\mu)$ is an optional regularization term encoding prior knowledge about parameter values.

*Case Study: Hidden Fluid Mechanics* (Raissi et al., 2020): Raissi et al. demonstrated parameter identification for the Navier-Stokes equations from scattered velocity measurements. Given noisy observations of the velocity field, the method simultaneously reconstructs the full flow field and infers the Reynolds number, achieving relative errors below 1% with measurements at only 0.1% of the domain points.

The key advantage: PINNs leverage physics to interpolate between sparse measurements, dramatically reducing data requirements compared to purely data-driven methods.

**2. Adaptive Sampling Strategies**

Uniform random sampling of collocation points often yields inefficient training, particularly in parametric settings where solution complexity varies across parameter space. Adaptive sampling concentrates points in regions of high PDE residual or parameter sensitivity.

*Residual-Based Adaptive Refinement (RAR):* At each training iteration, compute residuals at all collocation points and preferentially sample from high-residual regions (Daw et al., 2022; Gao et al., 2021; Lu et al., 2021b; Wu et al., 2020):

$$p(x, t, \mu) \propto |\mathcal{L}(u_\theta(x, t, \mu); \mu) - f(x, t; \mu)| \tag{24}$$

*Active Learning for Parameter Space:* For parametric problems, identify parameter values where the current model performs poorly and add training samples at those parameters. This mirrors the greedy algorithms of reduced basis methods but operates in a data-driven manner.

Recent work by Wu et al. (Wu et al., 2023a) demonstrated 50-70% reductions in required training data through parameter space active learning (Lookman et al., 2019; Wu et al., 2020) for Reynolds number-dependent flows.

**Advanced Training Methods (2023-2025):**

Advanced training methods have improved PINN performance. Wang et al. (Wang et al., 2025) introduced gradient alignment from a second-order optimization perspective, dynamically adjusting loss weights based on gradient alignment to achieve 30-50% training time reductions. De Ryck et al. (De Ryck et al., 2024) provided an operator preconditioning perspective on physics-informed training. Hwang and Lim (Hwang & Lim, 2024) developed dual cone gradient descent for handling conflicting gradients.

Hao et al. (Hao et al., 2024) released PINNacle at NeurIPS 2024, providing standardized PINN evaluation across 15 canonical PDEs with 20+ method variants. Miyagawa and Yokota (Miyagawa & Yokota, 2024) extended PINN theory to functional differential equations with convergence guarantees.

**3. Multi-Task and Transfer Learning**

Solving parametric PDEs inherently involves multiple related tasks (one per parameter value). Multi-task learning exploits this structure:

*Shared Representation Learning:* Train a single network on multiple parameter values simultaneously, encouraging the network to learn parameter-invariant features while maintaining parameter-specific outputs:

$$\mathcal{L}_{\text{MT}} = \sum_{i=1}^{N_{\text{tasks}}} w_i \mathcal{L}_{\text{total}}(\mu_i) \tag{25}$$

where $w_i$ are task-specific weights.

*Transfer Learning:* Pre-train on easy-to-solve parameter values, then fine-tune for challenging regions. For instance, train first on high-viscosity (smooth) solutions, then transfer to low-viscosity (sharper gradients) cases.

Desai et al. (Desai et al., 2021), along with comparative studies (Lu et al., 2022; Geneva & Zabaras, 2022), showed that transfer learning reduces training iterations by up to 80% when adapting PINNs across Reynolds numbers.

### 2.1.3 Recent Advances in PINN Architecture and Training

Recent developments have significantly improved PINN training efficiency, stability, and scalability through novel architectural designs and training strategies.

**Ill-Conditioning Analysis and Solutions** Cao & Zhang (2025) established a fundamental connection between PINN ill-conditioning and the Jacobian matrix condition number of the PDE system. By constructing controlled systems that adjust the condition number while preserving solutions, they demonstrated that reducing the Jacobian condition number leads to faster convergence and higher accuracy. This breakthrough enabled the first successful PINN simulation of three-dimensional flow around the M6 wing at Reynolds number 5,000, representing a significant milestone for industrial-complexity problems. Zhang et al. (2025b) built upon their foundation, applying it to the parametric domain.

**Separable Architectures** To mitigate the curse of dimensionality in multi-dimensional PDEs, Cho et al. (2023) introduced Separable Physics-Informed Neural Networks (SPINN). Operating on a per-axis basis rather than point-wise processing, SPINN dramatically reduces the number of network forward passes and memory overhead. By leveraging forward-mode automatic differentiation, SPINN enables training with over $10^7$ collocation points on a single GPU—orders of magnitude more than conventional PINNs can handle.

**Kolmogorov-Arnold Networks for PDEs** A paradigm shift in neural architecture comes from Kolmogorov-Arnold Networks (KAN), which replace fixed activation functions with learnable spline-based functions. Wang et al. (2024) proposed Kolmogorov-Arnold-Informed Neural Networks (KINN), systematically comparing KAN and MLP across various PDE formulations including strong form, energy form, and inverse form. KINN demonstrates significant advantages in accuracy and convergence for multi-scale, singularity, stress concentration, and nonlinear hyperelasticity problems, offering better interpretability with fewer parameters.

Building on this foundation, Jacob et al. (2024) introduced Separable Physics-Informed Kolmogorov-Arnold Networks (SPIKANs), applying the separation of variables principle to KANs. By decomposing problems such that each dimension is handled by an individual KAN, SPIKANs drastically reduce computational complexity without sacrificing accuracy, particularly effective for higher-dimensional PDEs where collocation points grow exponentially with dimensionality.

Most recently, Zhang et al. (2025c) proposed Legend-KINN, integrating Legendre orthogonal polynomials into the KAN architecture combined with pseudo-time stepping in backpropagation. Legend-KINN achieves 1–3 orders of magnitude faster convergence than both KAN and MLP under identical parameter settings. The orthogonality of Legendre polynomials ensures effective error distribution across dimensions while maintaining strict adherence to physical constraints, addressing the interpretability and stability issues of conventional PINNs.

#### 2.1.4 Advantages and Limitations: A Critical Analysis

**Strengths of PINNs for Parametric Problems:**

*1. Mesh-Free Formulation:* PINNs avoid mesh generation, making them naturally suited for complex and parameterized geometries. This is particularly valuable when geometry itself is a parameter.

*2. Data Efficiency Through Physics:* By encoding governing equations, PINNs can learn from sparse data—often orders of magnitude less than purely data-driven methods. This is crucial in scenarios where high-fidelity simulations are expensive.

*3. Inverse Problem Capability:* The ability to treat parameters as learnable variables enables powerful parameter identification frameworks, with direct applications in calibration and data assimilation.

*4. Flexible Uncertainty Quantification:* Bayesian extensions like B-PINNs (Yang et al., 2021) and related uncertainty quantification approaches (Yang & Perdikaris, 2019; Psaros et al., 2023; Lakshminarayanan et al., 2017; Tripathy & Bilionis, 2018) naturally quantify parameter and prediction uncertainties without requiring multiple forward solves.

*5. Multi-Physics Integration:* PINNs can naturally couple multiple physics through combined loss terms, useful for multi-scale parametric problems.

**Fundamental Limitations:**

*1. Parameter Space Generalization:* The most critical weakness for parametric problems: standard PINNs typically require retraining for each new parameter region. While a single network can handle modest parameter ranges during training, extrapolation to out-of-distribution parameters often fails catastrophically. A network trained on $\nu \in [0.01, 0.05]$ may produce nonsensical results for $\nu = 0.1$.

*2. Training Instability:* Balancing multiple loss terms in Eq. equation 13 remains challenging. Stiff PDEs, multi-scale problems, and high Reynolds number flows often exhibit training instabilities, with loss components competing rather than cooperating (Krishnapriyan et al., 2021; Wang et al., 2022a).

*3. Computational Cost:* Each training step requires evaluating PDE residuals at numerous collocation points, involving expensive automatic differentiation operations. For high-dimensional problems, training can take days to weeks on GPUs—comparable to or exceeding traditional solver costs for single parameter values.

*4. Convergence Guarantees:* Rigorous convergence theory remains incomplete. While universal approximation theorems suggest that neural networks can approximate PDE solutions arbitrarily well in principle, practical convergence rates and required network sizes are problem-dependent and often unpredictable.

*5. Spectral Bias:* Neural networks exhibit spectral bias toward learning low-frequency components of solutions (Rahaman et al., 2019). For parametric problems with high-frequency features (shocks, boundary layers) that vary with parameters, this bias necessitates very deep networks or specialized architectures.

**When to Use PINNs for Parametric Problems:**

PINNs excel in specific scenarios where their unique capabilities align with problem requirements. They are particularly effective for few-shot or single-query problems when solutions are needed for only a handful of parameter values, making the training cost per parameter acceptable. Their inverse problem capability makes them invaluable when parameters must be inferred from sparse observational data, leveraging physics to interpolate between measurements. In sparse data regimes where high-fidelity training data is scarce but governing physics is well-understood, PINNs can extract maximum information by encoding physical laws as constraints. Additionally, their mesh-free nature provides advantages for complex or parameterized geometries where traditional mesh generation becomes prohibitively expensive.

For multi-query parametric scenarios requiring rapid solution evaluation across broad parameter spaces, neural operators (discussed next) offer superior performance through their inherent ability to generalize across parameter distributions in a single training phase.

**Empirical Lesson:** "PINNs pioneered physics-informed learning but face challenges in parameter space generalization. Each new parameter region often requires retraining, limiting efficiency for parametric studies. This motivates the neural operator paradigm."

### 2.2 Neural Operators: Operator Learning for Parameter Space Generalization

In contrast to PINNs, which approximate solutions for specific parameter values, neural operators learn mappings between infinite-dimensional function spaces. This paradigm shift enables unprecedented parameter space generalization: a single trained neural operator can evaluate solutions across entire parameter distributions without retraining.

#### 2.2.1 Mathematical Foundations of Operator Learning

The theoretical foundation rests on extending universal approximation from functions to operators.

**Classical Setting:** Traditional neural networks approximate mappings $f : \mathbb{R}^n \to \mathbb{R}^m$ between finite-dimensional spaces.

**Operator Setting:** Neural operators approximate mappings $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ between infinite-dimensional function spaces, where: $\mathcal{A}$ is the input function space (e.g., space of initial conditions, coefficient functions, or parameter-dependent forcing), and $\mathcal{U}$ is the output solution space

For parametric PDEs, the solution operator $\mathcal{G}$ defined in Eq. equation 8 maps parameters and input functions to solutions. Neural operators aim to learn $\mathcal{G}_\theta \approx \mathcal{G}$.

**Universal Approximation for Operators:**

Chen and Chen (Chen & Chen, 1995) established that neural networks can approximate continuous operators arbitrarily well—a result generalized by Kovachki et al. (Kovachki et al., 2023a) for modern neural operator architectures.

**Theorem 1** (Neural Operator Universal Approximation (Informal))**.** *Let $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ be a continuous operator between function spaces. For any $\epsilon > 0$, there exists a neural operator architecture and parameters $\theta$ such that:*

$$\sup_{a \in \mathcal{A}} \|\mathcal{G}(a) - \mathcal{G}_\theta(a)\|_{\mathcal{U}} < \epsilon \tag{26}$$

This theoretical foundation justifies the neural operator approach: in principle, a sufficiently large network can learn the complete parameter-to-solution map.

**Discretization Invariance:**

A crucial advantage of neural operator formulations is discretization invariance: the operator $\mathcal{G}_\theta$ operates on continuous functions, not discrete grid representations. This means a neural operator trained on coarse-resolution data can evaluate solutions on fine-resolution grids—a capability called zero-shot super-resolution that has no traditional numerical methods analogue.

Formally, if $u_h^{\text{coarse}}$ and $u_h^{\text{fine}}$ represent discretizations of the same continuous solution $u$:

$$\mathcal{G}_\theta(a) \text{ can be evaluated at any discretization} \tag{27}$$

This property stems from operating in function space rather than grid space.

### 2.2.2 DEEPONET: DEEP OPERATOR NETWORKS

DeepONet, introduced by Lu et al. (Lu et al., 2021a) in *Nature Machine Intelligence*, represents the first practical implementation of operator learning achieving widespread success.

**Architecture: Branch-Trunk Decomposition**

The key innovation is decomposing the operator evaluation as:

$$\mathcal{G}_\theta(a)(y) = \sum_{k=1}^{p} b_k(a) \cdot t_k(y) \tag{28}$$

where: $a$ is an input function (e.g., initial condition, coefficient function, or parameter encoding) Moreover, $y$ is the evaluation location (spatial-temporal coordinate) Additionally, $b_k : \mathcal{A} \to \mathbb{R}$ are *branch networks* that encode the input function Furthermore, $t_k : \Omega \times [0, T] \to \mathbb{R}$ are *trunk networks* that encode the evaluation location Also, $p$ is the latent dimension (typically $p \sim 100 - 1000$)

The branch network takes as input a discretized representation of the input function $a$ evaluated at sensor locations $\{a(x_i)\}_{i=1}^m$:

$$\mathbf{b} = [b_1(a), \ldots, b_p(a)] = \text{BranchNet}([a(x_1), \ldots, a(x_m)]) \tag{29}$$

The trunk network takes the query location $y$:

$$\mathbf{t} = [t_1(y), \ldots, t_p(y)] = \text{TrunkNet}(y) \tag{30}$$

The final solution is computed via dot product:

$$\mathcal{G}_\theta(a)(y) = \mathbf{b}^\top \mathbf{t} = \sum_{k=1}^{p} b_k(a) \cdot t_k(y) \tag{31}$$

**Parametric PDEs with DeepONet:**

For parametric problems, parameters $\mu$ can be incorporated in multiple ways:

*Approach 1: Parameters as Branch Input*

$$\mathbf{b} = \text{BranchNet}([a(x_1), \ldots, a(x_m), \mu_1, \ldots, \mu_d]) \tag{32}$$

Parameters are appended to the function sensors in the branch network input.

*Approach 2: Parameters as Trunk Input*

$$\mathbf{t} = \text{TrunkNet}([y, \mu]) \tag{33}$$

Parameters are concatenated with query coordinates.

*Approach 3: Dual Encoding* Both networks receive parameter information, enabling maximum expressivity:

$$\mathbf{b} = \text{BranchNet}([a(x_1), \ldots, a(x_m), \mu]) \tag{34}$$

$$\mathbf{t} = \text{TrunkNet}([y, \mu]) \tag{35}$$

**Physics-Informed DeepONet (PI-DeepONet)**

Wang et al. (Wang et al., 2021a) extended DeepONet with physics-informed training, combining data and physical constraints. The loss function:

$$\mathcal{L}_{\text{PI-DeepONet}} = \mathcal{L}_{\text{data}} + \lambda_{\text{PDE}} \mathcal{L}_{\text{PDE}} \tag{36}$$

where:

$$\mathcal{L}_{\text{data}} = \mathbb{E}_{a,y}[\|\mathcal{G}_\theta(a)(y) - \mathcal{G}(a)(y)\|^2] \tag{37}$$

$$\mathcal{L}_{\text{PDE}} = \mathbb{E}_{a,y}[\|\mathcal{L}(\mathcal{G}_\theta(a)(y); \mu) - f(y; \mu)\|^2] \tag{38}$$

This hybrid approach offers remarkable data efficiency: PI-DeepONet can learn operators from as few as 5-10 solution snapshots by leveraging PDE residuals to regularize the learned mapping.

**Case Study: Parametric Burgers Equation**

Consider learning the solution operator for Burgers equation with parametric viscosity and initial conditions:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \quad u(x,0) = a(x) \tag{39}$$

where $\nu \in [0.01, 0.1]$ and $a(x)$ varies over a function space (e.g., random Gaussian processes).

*Setup:* Input function: $a(x)$ sampled at $m = 100$ sensor locations, Parameters: viscosity $\nu$, and Output: $u(x,t)$ for any $(x,t)$

*Training:* Generate $N = 1000$ training triplets $(a_i, \nu_i, u_i)$ where $u_i$ solves Burgers equation with IC $a_i$ and viscosity $\nu_i$, Branch network: MLP with input $[a_i(x_1), \ldots, a_i(x_m), \nu_i] \in \mathbb{R}^{101}$, output $\mathbf{b}_i \in \mathbb{R}^{100}$, Trunk network: MLP with input $(x,t) \in \mathbb{R}^2$, output $\mathbf{t} \in \mathbb{R}^{100}$, and Minimize: $\mathcal{L} = \sum_{i,j} |\mathbf{b}_i^\top \mathbf{t}_j - u_i(x_j, t_j)|^2$

*Results:* Once trained, the DeepONet evaluates solutions for any new $(a, \nu)$ in $\sim$1ms on GPU, achieving relative $L^2$ errors of 1-3% across the parameter range—including parameter values never seen during training. This demonstrates true parameter space generalization.

**Variants and Extensions:**

*POD-DeepONet:* Bhattacharya et al. (Bhattacharya et al., 2021) combined proper orthogonal decomposition with DeepONet. The branch network predicts POD coefficients, and the trunk network reconstructs the solution from basis functions. This reduces output dimensionality and improves data efficiency.

*MIONet (Multiple-Input Operator Network):* Extends DeepONet to multiple input functions, enabling multi-physics or multi-fidelity (Penwarden et al., 2023; Howard et al., 2023; Peherstorfer et al., 2018) learning (Jin et al., 2022a).

*Fourier-DeepONet:* Incorporates Fourier feature embeddings in the trunk network to better capture high-frequency solution components (Li et al., 2020b).

**Advanced DeepONet Variants:**

Recent developments have transformed DeepONet methodology. Mandl et al. (Mandl et al., 2024) introduced Separable Physics-Informed DeepONet, addressing the curse of dimensionality through separable architecture combined with forward-mode automatic differentiation. This achieved linear scaling with dimensionality—for a 4D heat equation, training time reduced from 289 hours to 2.5 hours, a remarkable 100$\times$ speedup. Zhang et al. (Zhang et al., 2024) developed PAR-DeepONet with physical adaptive refinement, dynamically adjusting sampling based on PDE residuals and achieving up to 71.3% accuracy improvements.

A breakthrough came from Feng et al. (Feng et al., 2025b) in Nature Communications, demonstrating one-shot operator learning from single solution trajectories using self-supervised learning and meta-learning, achieving 5-10% relative errors with just one example—previously requiring hundreds. This is revolutionary for expensive simulations. Yang (Yang, 2025) provided comprehensive generalization bounds for DeepONet with physics-informed training, deriving error bounds in terms of Rademacher complexity. Li et al. (Li et al., 2025a) analyzed the trunk-branch architecture and proposed extensions allowing nonlinear interference for improved performance on nonlinear parametric PDEs. Zhong et al. (Zhong & Meidani, 2024) introduced physics-informed compositional DeepONet for complex multi-physics problems with hierarchical decomposition.

### 2.2.3 Fourier Neural Operator (FNO)

The Fourier Neural Operator, introduced by Li et al. (Li et al., 2021) at ICLR 2021, represents a breakthrough in operator learning through spectral methods. FNO achieves remarkable computational efficiency and zero-shot super-resolution capabilities by operating in the Fourier domain, making it particularly effective for parametric PDEs on periodic domains or problems amenable to Fourier analysis.

**Motivation: Why the Frequency Domain?**

Traditional convolutional neural networks have limited receptive fields, requiring many layers to capture long-range dependencies. PDE solutions often exhibit multi-scale structures where distant spatial locations couple through the governing equations. Operating in the Fourier domain offers three key advantages:

*1. Global receptive field:* Every Fourier mode couples with all spatial locations after inverse transform, enabling immediate global information propagation.

*2. Efficient convolution:* The convolution theorem states that convolution in physical space is multiplication in Fourier space, enabling $\mathcal{O}(N \log N)$ operations via Fast Fourier Transform (FFT) rather than $\mathcal{O}(N^2)$.

*3. Multi-scale representation:* Low and high frequencies naturally encode coarse and fine solution features, respectively, facilitating multi-scale learning.

**Architecture: Spectral Convolution Layers**

The FNO architecture consists of multiple Fourier layers. Each layer transforms the input through spectral convolution:

$$v_{k+1}(x) = \sigma\left(W \cdot v_k(x) + (\mathcal{K} \cdot v_k)(x)\right) \tag{40}$$

where $v_k$ is the feature field at layer $k$, $W$ is a standard pointwise linear transformation, $\mathcal{K}$ is the spectral convolution operator, and $\sigma$ is a nonlinear activation (typically GELU).

The spectral convolution is the key innovation:

$$(\mathcal{K} \cdot v_k)(x) = \mathcal{F}^{-1}\left(R \cdot \mathcal{F}(v_k)\right)(x) \tag{41}$$

where $\mathcal{F}$ denotes the Fourier transform, $R(\omega) \in \mathbb{C}^{d_v \times d_v}$ is a learnable complex-valued weight in frequency space, $\omega$ represents frequency modes, and $\mathcal{F}^{-1}$ is the inverse Fourier transform.

Crucially, only low-frequency modes are retained (typically the first $k_{\max}$ modes in each dimension), acting as a learned low-pass filter that preserves essential solution features while discarding high-frequency noise.

**Parametric FNO: Incorporating Parameters**

For parametric PDEs, three main strategies integrate parameters into FNO:

*Strategy A: Parameter-Modulated Spectral Filters*

$$R(\omega; \mu) = \text{MLP}(\mu) \odot R_{\text{base}}(\omega) \tag{42}$$

A multi-layer perceptron (MLP) generates parameter-dependent scaling factors that modulate the base spectral filters. This allows different frequencies to be emphasized or suppressed based on parameter values—for instance, low-frequency modes for high-viscosity flows.

*Strategy B: Parameter as Initial Channel*

$$v_0(x) = [\text{Lift}(a(x)), \mu_1, \ldots, \mu_d] \tag{43}$$

Parameters are broadcast across the spatial domain and concatenated as additional input channels. A lifting layer Lift projects the input function $a(x)$ to higher-dimensional feature space before concatenating with parameters.

*Strategy C: Multi-Task Architecture* Multiple FNO branches are trained simultaneously, with shared low-level layers and parameter-specific high-level layers, enabling efficient multi-query training.

**Geometric FNO (Geo-FNO)**

Standard FNO assumes periodic domains and regular grids. Li et al. (Li et al., 2023b) extended FNO to irregular geometries through several techniques:

*1. Domain mapping:* Map irregular physical domain $\Omega(\mu)$ to a regular computational domain $\hat{\Omega}$ via diffeomorphisms $\Phi_\mu : \Omega(\mu) \to \hat{\Omega}$. The PDE is then solved in $\hat{\Omega}$ where standard Fourier methods apply.

*2. Non-uniform FFT:* For non-periodic or irregular domains, use type-3 non-uniform FFT (NUFFT) that handles scattered data points.

*3. Geometric encoding:* Embed geometric information (e.g., distance fields, curvature) as additional input channels to inform the network about domain shape.

These advances enable FNO application to geometry-parameterized problems, such as airfoil shape optimization where the domain boundary varies with parameters.

**Physics-Informed FNO (PI-FNO)**

Wang et al. (Wang et al., 2022a) combined FNO with physics-informed training:

$$\mathcal{L}_{\text{PI-FNO}} = \mathcal{L}_{\text{data}} + \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} \tag{44}$$

The PDE residual loss is computed by differentiating the FNO prediction:

$$\mathcal{L}_{\text{PDE}} = \mathbb{E}\left[\left|\mathcal{L}(\mathcal{G}_\theta(a)(x,t);\mu) - f(x,t;\mu)\right|^2\right] \tag{45}$$

Automatic differentiation computes spatial derivatives of FNO outputs efficiently due to the spectral representation—derivatives in Fourier space are simply multiplication by $i\omega$.

**Case Study: Navier-Stokes at Varying Reynolds Numbers**

Li et al. (Li et al., 2021), building on earlier work in turbulence modeling (Wang et al., 2017; Kurth et al., 2023; List et al., 2023), demonstrated FNO on 2D turbulent flows governed by the Navier-Stokes vorticity equation:

$$\frac{\partial \omega}{\partial t} + u \cdot \nabla \omega = \frac{1}{\text{Re}} \nabla^2 \omega + f, \quad \nabla \cdot u = 0 \tag{46}$$

*Setup:* The experiments consider turbulent flows with Reynolds numbers ranging from 1000 to 10000. The network takes as input the initial vorticity $\omega_0(x)$ and random forcing $f(x,t)$, and outputs the vorticity field $\omega(x,t)$ at future times. Training data consists of 1000 direct numerical simulations on $256 \times 256$ grids. The FNO architecture employs 4 Fourier layers, keeping the 12 lowest modes per dimension.

*Results:* The trained FNO achieves a relative $L^2$ error of 1.8% averaged across Reynolds numbers, including unseen Re values through both interpolation and modest extrapolation. Inference time is remarkably fast at 0.005s per query on GPU compared to 5-10 minutes for DNS, yielding a speedup of approximately 60,000×. The model exhibits zero-shot super-resolution capability, generalizing from training on $64 \times 64$ data to $256 \times 256$ evaluation with less than 5% error increase. Furthermore, autoregressive rollout for $t \in [0, 50]$ maintains physical properties including energy conservation and enstrophy cascade, demonstrating long-time stability.

This demonstrates FNO's capability for multi-query parametric scenarios: one training session enables rapid exploration of the Reynolds number space.

**FNO Enhancements:**

FNO has been enhanced through refinements addressing computational efficiency and accuracy. Li et al. (Li et al., 2025d) introduced the Decomposed Fourier Neural Operator (D-FNO) in CMAME 2025, leveraging tensor decomposition to reduce 3D complexity from $O(N^3 \log N)$ to $O(PN \log N)$, achieving 2-3× speedup over Factorized FNO while maintaining accuracy. Qin et al. (Qin et al., 2024) developed SpecBoost-FNO addressing frequency bias through ensemble learning, achieving 50% average accuracy improvement across benchmark problems. Kong et al. (Kong et al., 2025) applied spectral-boosted FNO to 3D seismic wavefield modeling, reducing frequency bias by 40%.

For inverse problems, Feng et al. (Feng et al., 2025a) introduced Sensitivity-Constrained FNO (SC-FNO) at ICLR 2025, integrating sensitivity analysis into the operator framework for improved parameter inversion. Zhang et al. (Zhang et al., 2025a) developed physics-informed FNO with enhanced constraint enforcement mechanisms.

Conservation laws received special attention: Cardoso-Bihlo and Bihlo (Cardoso-Bihlo & Bihlo, 2025) developed exactly conservative physics-informed neural operators in Neural Networks 2025, ensuring discrete conservation of mass, momentum, and energy through learnable adaptive correction. Liu and Tang (Liu & Tang, 2025) combined diffusion models with FNO (DiffFNO) at CVPR 2025 for better uncertainty quantification and robustness, achieving 15-25% accuracy improvements. Kalimuthu et al. (Kalimuthu et al., 2025) proposed LOGLO-FNO at ICLR 2025 for efficient local-global feature learning. Loeffler et al. (Loeffler et al., 2025) provided rigorous error analysis for FNO on parametric PDEs, establishing convergence rates.

For variable geometries, Zhong and Meidani (Zhong & Meidani, 2025) developed Physics-Informed Geometry-Aware Neural Operator (PI-GANO) in CMAME 2025, simultaneously generalizing across PDE parameters and domain geometries using signed distance functions and parameter-geometry attention mechanisms, achieving ¡3% relative errors across diverse geometric configurations without expensive FEM data generation.

**Comparative Analysis: FNO vs DeepONet**

Table 2 compares the two dominant neural operator architectures for parametric PDEs.

**Performance Insights:** FNO excels when solutions have strong spectral content and domains are regular Moreover, DeepONet is preferable for complex geometries or when input functions are sparsely sampled

Table 2: Comparison of FNO and DeepONet for parametric PDE solving

| Aspect | FNO | DeepONet |
|---|---|---|
| **Core Mechanism** | Spectral convolution in Fourier space | Branch-trunk factorization |
| **Parameter Generalization** | Strong (especially for smooth parameter dependence) | Strong (flexible parameter encoding) |
| **Training Data** | High (1000-10000 samples) | Moderate (100-1000 samples) |
| **Inference Speed** | Fastest ($\mathcal{O}(N \log N)$ via FFT) | Fast ($\mathcal{O}(N_{\text{query}})$) |
| **Complex Geometry** | Moderate (requires Geo-FNO or NUFFT) | Strong (mesh-independent) |
| **Zero-shot Super-resolution** | Native capability | Limited (requires sensor distribution design) |
| **Physics Constraints** | Optional via PI-FNO | Natural via PI-DeepONet |
| **Best Use Cases** | Periodic/regular domains, turbulent flows, time-dependent evolution | Irregular geometries, sparse sensors, inverse problems |

Additionally, For high Reynolds number flows, FNO's spectral bias toward smooth functions can be problematic; hybrid approaches help Furthermore, Data efficiency: PI-DeepONet often requires 5-10$\times$ less data than FNO by leveraging physics Also, Computational cost: FNO training is faster per epoch but may need more epochs for convergence

### 2.2.4 Graph Neural Operators and Advanced Architectures

While DeepONet and FNO dominate the neural operator landscape, several emerging architectures address specific limitations or application scenarios.

**Graph Neural Operator (GNO)**

Proposed by Li et al. (Li et al., 2020a), with extensions in (Li et al., 2023a; Hao et al., 2023b; Gao et al., 2022; Pfaff et al., 2021; Sanchez-Gonzalez et al., 2020), GNO represents functions on graphs rather than regular grids, naturally handling unstructured meshes arising in finite element simulations or point clouds from experimental measurements.

*Key idea:* Represent the function $a(x)$ as a graph signal on vertices $\{v_i\}$: $a = [a(v_1), \ldots, a(v_N)]$. Graph convolutional layers learn the operator:

$$[\mathcal{G}_\theta(a)]_i = \sigma\left(\sum_{j \in \mathcal{N}(i)} W_{ij} a_j + b_i\right) \tag{47}$$

where $\mathcal{N}(i)$ denotes neighbors of vertex $i$.

*Parametric extension:* For geometry-parameterized problems, the graph structure itself varies with $\mu$. GNO handles this by learning message-passing rules invariant to graph topology. Parameters can modulate edge weights: $W_{ij}(\mu) = \text{MLP}(\mu, \|v_i - v_j\|, \text{edge features})$.

*Applications:* Particularly effective for finite element meshes in structural mechanics, where geometry changes significantly (e.g., topology optimization, crack propagation with varying paths).

**Graph-Based Extensions:**

Graph neural operators have seen significant methodological advances for challenging geometries. Liao et al. (Liao et al., 2025) developed curvature-aware graph attention at ICML 2025 for PDEs on manifolds, incorporating Riemann curvature tensors into message passing for accurate differential equation solving on curved surfaces. Lino et al. (Lino et al., 2025) introduced diffusion graph networks at ICLR 2025 for complex fluid simulations with irregular boundaries, using learned diffusion processes on graphs. Zou et al. (Zou et al., 2025) proposed finite-difference-informed graph networks in Physics of Fluids 2025 for incompressible flows on block-structured grids, elegantly combining GNN flexibility with finite-difference structure.

**Multipole Graph Neural Operator (MGNO)**

Li et al. (Li et al., 2023c) introduced MGNO to efficiently handle long-range interactions in graph-based operators. The key innovation is a multipole expansion separating near-field (local) and far-field (global) interactions:

$$[\mathcal{G}_\theta(a)]_i = \text{Local}(a, \mathcal{N}_{\text{near}}(i)) + \text{Global}(a, \text{multipole coefficients}) \tag{48}$$

This reduces computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$ for large-scale problems, enabling neural operators on million-node meshes.

**U-Net Based Operators**

Gupta and Brandstetter (Gupta & Brandstetter, 2022) adapted the U-Net encoder-decoder architecture for operator learning. The multi-scale nature of U-Nets—with skip connections preserving fine details—makes them effective for multi-scale parametric PDEs.

$$\mathcal{G}_\theta(a)(x) = \text{Decoder}_\theta(\text{Encoder}_\theta(a))(x) \tag{49}$$

*Parametric integration:* Parameters influence multiple scales through: *Encoder conditioning:* $\text{Encoder}_\theta(a; \mu)$ modulates downsampling based on $\mu$, *Bottleneck injection:* Inject $\mu$ at the U-Net bottleneck where global information is processed, and *Decoder conditioning:* Parameter-dependent upsampling for reconstruction

*Advantage:* U-Net's hierarchical structure naturally handles problems where parameters control multiple length scales (e.g., turbulent flows where Re affects both large eddies and small dissipative scales).

**Transformer-Based Operators**

Recent work (Cao, 2021; Hao et al., 2023a) explores attention mechanisms for operator learning. The Transolver (Wu et al., 2023b; Cao et al., 2023) architecture uses:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{50}$$

where queries $Q$, keys $K$, and values $V$ are derived from the input function and parameters.

*Parameter-aware attention:* Parameters modulate attention weights:

$$\text{Attention}(Q, K, V; \mu) = \text{softmax}\left(\frac{QK^T + B(\mu)}{\sqrt{d_k}}\right)V \tag{51}$$

where $B(\mu)$ is a parameter-dependent bias learned by an MLP.

*Benefits:* Transformers capture long-range dependencies without the periodicity requirements of FNO, and their flexibility accommodates diverse parameter types (physical, geometric, boundary conditions) in a unified framework.

**Alternative Architectures:**

Emerging paradigms explore fundamentally new approaches to operator learning. Zheng et al. (Zheng et al., 2024) introduced the Mamba Neural Operator at NeurIPS 2024, applying state-space models to PDEs. Mamba uses adaptive state-space matrices with alias-free design for capturing long-range spatiotemporal dependencies at reduced computational cost compared to transformers, achieving competitive performance with better scaling.

Memory-enhanced models address long-time integration challenges. Buitrago-Restrepo et al. (Buitrago et al., 2025) demonstrated explicit memory mechanisms for time-dependent PDEs at ICLR 2025. By maintaining memory buffers storing spatiotemporal patterns, they significantly reduced autoregressive errors for long-time integration (500 vs 100 stable time steps, 40% lower error). Le Boudec et al. (Le Boudec et al., 2025) and Moro et al. (Moro & Chamon, 2025) proposed novel attention mechanisms specifically designed for operator learning. Li et al. (Li et al., 2025b) introduced max-up training strategies that improve out-of-distribution generalization by augmenting training with worst-case parameter perturbations.

*Challenges:* Quadratic complexity in the number of discretization points limits scalability; hybrid local-global attention mechanisms mitigate this.

### 2.3 Method Comparison and Selection Guidelines

Having reviewed the major methodologies, we now provide practical guidance for selecting appropriate methods based on problem characteristics and requirements.

### 2.3.1 METHOD CHARACTERISTICS

Different neural PDE methods have distinct characteristics suited for specific scenarios:

**Parameter Generalization:** Neural operators (DeepONet, FNO, GNO) learn mappings that generalize across parameter spaces without retraining, while PINNs typically require retraining for each parameter region.

**Data Requirements:** Physics-informed methods (PINNs, PI-DeepONet) can work with sparse or no paired training data by leveraging PDE residuals. Data-driven operators (vanilla FNO, DeepONet) require hundreds to thousands of high-fidelity solutions.

**Computational Cost:**

- FNO inference: $\mathcal{O}(N \log N)$ via FFT
- DeepONet inference: $\mathcal{O}(N_{\text{query}} \cdot d_{\text{latent}})$
- PINN inference: $\mathcal{O}(N_{\text{collocation}} \cdot \text{network depth})$

**Geometric Flexibility:** Graph neural operators handle arbitrary mesh topologies naturally. PINNs work well on irregular domains. FNO excels on regular/periodic domains but requires extensions (Geo-FNO) for complex geometries.

**Evaluation Criteria Definitions:** *Parameter Generalization:* Ability to accurately predict solutions for parameter values outside the training distribution *Training Data:* Number of high-fidelity simulations required (Low: ¡100, Medium: 100-1000, High: ¿1000) *Training Time:* Wall-clock time on modern GPUs (Short: ¡1 hour, Medium: 1-10 hours, Long: ¿10 hours) *Inference:* Query time per new parameter (Slow: ¿1s, Fast: 0.01-1s, Very Fast: ¡0.01s) *Complex Geometry:* Handling of irregular, parameterized, or time-varying domains *High-Dim Parameters:* Performance when $d > 50$

### 2.3.2 METHOD SELECTION GUIDE

The following text outlines the evaluation steps for selecting an appropriate method:

**Step 1: Data Availability Assessment**

*Question:* How much training data is available or can be generated?

**Abundant data (¿1000 samples):** Consider pure data-driven approaches Regular domain → *FNO* (fastest inference), and Irregular domain → *GNO* or *DeepONet* **Moderate data (100-1000 samples):** Use hybrid physics-data methods Multiple parameter queries expected → *PINO* or *PI-DeepONet*, and Single or few queries → *PINN* with transfer learning **Scarce data (¡100 samples):** Physics known → *PINN* or *PI-DeepONet*, and Physics unknown → Traditional methods or experimental design for more data

**Step 2: Query Pattern Analysis**

*Question:* How many parameter configurations will be queried?

**Single query:** PINN or traditional solver (no need for operator learning) **Few queries (2-10):** PINN with transfer learning or multi-task PINN **Many queries (¿10):** Neural operators essential Real-time requirements (ms inference) → *FNO* (if geometry allows) or *L-DeepONet*, and Standard requirements (sub-second) → *DeepONet*, *PINO*, or *Geo-FNO*

**Step 3: Geometric Complexity**

*Question:* What is the nature of the spatial domain?

**Regular/periodic:** FNO (optimal), **Fixed irregular:** Geo-FNO or GNO, **Parameterized geometry:** GNO or PINN, and **Time-varying geometry:** GNO with dynamic graphs or moving-mesh PINN

**Step 4: Parameter Dimensionality**

*Question:* How many parameters ($d$)?

**Low ($d < 10$):** All methods applicable **Medium ($d = 10$-$50$):** Neural operators with active subspace methods **High ($d > 50$):** Dimensionality reduction (active subspaces, POD) + neural operators, L-DeepONet with latent parameter encoding, and Consider whether all parameters are truly active (sensitivity analysis)

**Step 5: Physical Constraints Importance**

*Question:* Are conservation laws, symmetries, or boundary conditions critical?

**Critical (safety, physical validity):** Physics-informed methods PINN (hard-codes physics), and PI-DeepONet or PINO (soft physics constraints) **Important but flexible:** PINO with physics loss weighting **Not critical (pure prediction):** Data-driven FNO, DeepONet, or GNO

### 2.3.3 Application-Specific Recommendations

We conclude this section with concrete recommendations for common parametric PDE scenarios:

**Scenario 1: Airfoil Optimization (Reynolds Number and Shape Parameters)**

*Characteristics:* $d \sim$ 5-10 (Re + shape parameters), complex geometry, many design evaluations needed, CFD data expensive.

*Recommendation:* Geo-FNO or GNO Generate $\sim$500-1000 CFD simulations covering parameter space Moreover, Use Geo-FNO if shape parameterization is smooth (B-splines, NURBS) Additionally, Use GNO if meshes vary significantly or topology changes Furthermore, Expected speedup: 1000-10,000$\times$ vs CFD per query Also, Accuracy: 2-5% relative error typical

*Alternative:* PINO if data budget is limited to ¡200 samples, leveraging physics to compensate.

**Scenario 2: Parameter Identification from Sparse Measurements**

*Characteristics:* Unknown material properties, few sensors, single or few parameter sets of interest, physics well-understood.

*Recommendation:* PINN Formulate as inverse problem with learnable parameters, Leverage physics constraints to interpolate between sensors, Can work with as few as 5-10 sensors, and Uncertainty quantification via B-PINN

*Rationale:* Neural operators require abundant data; PINN's physics encoding compensates for data scarcity in single-query scenarios.

**Scenario 3: Uncertainty Quantification with 100+ Parameters**

*Characteristics:* High-dimensional parameter space (material uncertainties, manufacturing tolerances), need for probability distributions of quantities of interest, many MC samples required.

*Recommendation:* Active subspace + DeepONet or PINO Perform sensitivity analysis to identify active subspace (typically $d_{\text{eff}} \sim$ 5-10 even if nominal $d > 100$), Train neural operator on reduced parameter space, Use trained operator for MC sampling (millions of cheap evaluations), and Validate with few full-fidelity samples

*Alternative:* Sparse grid collocation if parameter dependence is smooth and $d < 30$.

**Scenario 4: Medical Imaging and Patient-Specific Modeling**

*Characteristics:* Each patient has unique geometry (anatomy), need for rapid diagnosis, limited data per patient, large population dataset available.

*Recommendation:* DIMON (Diffeomorphic Mapping Operator Network) or GNO Train on population of anatomies, Use diffeomorphic mapping to canonical domain (DIMON), Or directly use graph neural operators on patient meshes (GNO), and Enables real-time patient-specific simulations

*Example:* Yin et al. (Yin et al., 2024) demonstrated 1006 cardiac geometries with ¡2% error in electrophysiology simulations.

**Key Takeaway:** Method selection critically depends on the interplay between data availability, query frequency, geometric complexity, and parameter dimensionality. Hybrid physics-data approaches (PINO, PI-DeepONet) often provide the best balance, combining the data efficiency of physics-informed methods with the parameter generalization of neural operators.

# 3 Applications Across Scientific Domains

Having established the methodological foundations, we now examine how physics-informed neural networks and neural operators are transforming computational science across major application domains. This section demonstrates that the promise of rapid parameter space exploration translates to practical impact in fluid dynamics, structural mechanics, heat transfer, and electromagnetics. We emphasize parametric aspects throughout: how parameters enter each problem class, what computational challenges they pose, and how neural methods achieve breakthroughs.

## 3.1 Fluid Dynamics

Fluid dynamics presents some of the most compelling applications for parametric neural PDE solvers. The governing Navier-Stokes equations involve natural parameters (Reynolds, Mach, and Péclet numbers) that control flow regimes, while engineering applications demand parameter sweeps for design optimization. The computational expense of traditional CFD—particularly for turbulent flows—makes neural operators especially attractive.

### 3.1.1 Incompressible Navier-Stokes Equations

The incompressible Navier-Stokes equations govern fluid motion at low Mach numbers:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{\text{Re}}\nabla^2 \mathbf{u} + \mathbf{f}, \tag{52}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{53}$$

where $\mathbf{u}$ is velocity, $p$ is pressure, Re is the Reynolds number, and $\mathbf{f}$ represents body forces. The Reynolds number $\text{Re} = UL/\nu$ (characteristic velocity $\times$ length / kinematic viscosity) is a fundamental parameter controlling the flow regime from laminar ($\text{Re} < 2000$) through transitional to turbulent ($\text{Re} > 4000$).

**Parametric Challenges:** As Reynolds number increases, solutions develop increasingly fine-scale structures. Turbulent flows at $\text{Re} \sim 10^6$ exhibit energy cascades spanning orders of magnitude in scale, requiring extremely high resolution ($10^9$-$10^{12}$ degrees of freedom) for direct numerical simulation. Sweeping across Reynolds numbers to understand transition or optimize designs becomes computationally prohibitive.

**Neural Operator Breakthroughs:**

*1. Cylinder Flow Benchmark* (Jin et al., 2021): Jin et al. developed NSFnets (Navier-Stokes Flow nets) combining residual networks with physics constraints for flow around circular cylinders parameterized by Reynolds number and cylinder diameter.

*Problem Setup:* Parameters: $\text{Re} \in [40, 200]$, cylinder diameter $d \in [0.5, 1.5]$, Governing equations: Incompressible NS equation 52-equation 53, and Output: Velocity field $\mathbf{u}(x, y, t; \text{Re}, d)$ and pressure $p(x, y, t; \text{Re}, d)$

*Methodology:* Physics-informed neural networks with hard enforcement of boundary conditions on cylinder surface and adaptive collocation point sampling based on residual magnitude.

*Results:* Training on 200 CFD simulations covering the parameter space, NSFnets achieve relative $L^2$ error below 5% for velocity and below 8% for pressure. This translates to approximately 1000$\times$ speedup compared to CFD per query, reducing solution time from 15 minutes to 1 second. However, performance degrades for $\text{Re} > 200$ as the flow enters the transitional regime.

*Key Insight:* The vortex shedding frequency and Strouhal number ($St = fd/U$) predictions matched experimental correlations within experimental uncertainty, demonstrating physical fidelity.

*4. Advanced Fluid Applications:* The field has matured with production-grade applications. Qiu et al. (Qiu et al., 2025) performed direct numerical simulations of 3D two-phase flow using physics-informed neural networks with distributed parallel training on Journal of Fluid Mechanics 2025, handling millions of degrees of freedom for bubble dynamics and interfacial phenomena. Li et al. (Li et al., 2024a; 2025c) developed finite volume-informed neural networks (Physics of Fluids 2024) and finite element-informed networks (JCP 2025) for unsteady incompressible flows, achieving data-free approaches that rely purely on physics residuals without any simulation data. Zou et al. (Zou et al., 2025) integrated finite-difference schemes into graph networks in Physics of Fluids 2025 for flows on block-structured grids, achieving state-of-the-art accuracy for complex industrial geometries.

*2. FNO for Kolmogorov Flow* (Li et al., 2021): Li et al. applied Fourier Neural Operator to 2D forced turbulence, demonstrating unprecedented efficiency for parametric turbulent flow.

*Problem Setup:* The study employs the vorticity formulation $\partial_t \omega + \mathbf{u} \cdot \nabla \omega = \mathrm{Re}^{-1} \nabla^2 \omega + \nabla \times \mathbf{f}$ with Reynolds number $\mathrm{Re} \in [1000, 10000]$ and forcing frequency as parameters, using a $256 \times 256$ spatial grid resolution.

*Results (detailed):* Training on 1,000 DNS trajectories at various Re values, the FNO architecture with 4 Fourier layers retaining 12 lowest frequency modes achieves 1.8% relative $L^2$ error averaged across test Re values. The model demonstrates remarkable autoregressive stability over 50 time units while preserving the physical energy spectrum. Its zero-shot super-resolution capability allows training on $64 \times 64$ grids while evaluating at $256 \times 256$ with less than 3% error increase. The computational gain is dramatic, achieving 60,000$\times$ speedup with 0.005s inference time compared to 5-10 minutes per timestep for DNS.

*Physical Validation:* The learned operator correctly captures the Kolmogorov $k^{-5/3}$ energy spectrum in the inertial range, reproduces the enstrophy cascade to small scales, and properly represents the Reynolds number dependence of integral length scales.

*3. Cavity Flow with DeepONet* (Dong & Li, 2022): Dong et al. addressed lid-driven cavity flow—a canonical benchmark in CFD—using DeepONet with multi-fidelity (Penwarden et al., 2023; Howard et al., 2023; Peherstorfer et al., 2018) data fusion.

*Problem Setup:* Parameters: Reynolds number $\mathrm{Re} \in [100, 1000]$, cavity aspect ratio $L/H \in [1, 2]$, lid velocity profile $u_{\text{lid}}(x)$, and Challenge: Recirculation vortices whose number and position depend sensitively on Re

*Innovation:* Combined high-fidelity CFD data (expensive, $N = 50$ samples) with low-fidelity data (cheap, $N = 500$ samples) through a bi-fidelity DeepONet architecture where branch networks at different fidelity levels share trunk network weights.

*Results:* Relative error: $< 8\%$ using multi-fidelity (Penwarden et al., 2023; Howard et al., 2023; Peherstorfer et al., 2018) approach vs $> 15\%$ with high-fidelity-only training on same budget, Captured critical Re transition where secondary vortices appear, and Inference: 0.1s per parameter configuration

**Comparative Performance Summary:**

Table 3: Performance comparison of neural methods for parametric incompressible flows

| Work | Method | Param Dim | Re Range | Training Data | Inference | Rel. Error |
|---|---|---|---|---|---|---|
| Jin (2021) | NSFnets | 2 | 40-200 | 200 CFD | $\sim$1s | $< 5\%$ |
| Li (2021) | FNO | 1 | 1k-10k | 1000 DNS | 0.005s | 1.8% |
| Dong (2022) | DeepONet | 3 | 100-1k | 50 HF + 500 LF | 0.1s | $< 8\%$ |
| Traditional | CFD | N/A | Per Re | 1 solve | 10min-2hr | Reference |

### 3.1.2 Compressible Flows and Shocks

Compressible flows introduce additional parameters (Mach number, specific heat ratio) and computational challenges from discontinuities (shocks, contact discontinuities).

*Parametric Dimensions:* Mach number $M = U/c$ (velocity / sound speed), ratio of specific heats $\gamma$, geometry parameters.

*Neural Method Challenges:* Discontinuities violate smoothness assumptions underlying neural network approximation. Standard PINNs and neural operators struggle with Gibbs phenomena near shocks.

*Recent Progress:* Several approaches address shock capturing: *Conservative formulation PINNs* (Jagtap & Karniadakis, 2020): Enforce conservation laws in integral form to handle weak solutions, *Shock-capturing neural operators* (List et al., 2023): Augment FNO with WENO-inspired local refinement near discontinuities, and *Hyperbolic PINNs* (Moseley et al., 2023b): Domain decomposition along characteristic lines

*Example Application - Transonic Airfoil:* Mao et al. (Mao et al., 2023) used conditional neural operators for transonic flow over airfoils with varying Mach numbers ($M \in [0.7, 0.85]$, transonic regime) and angles of

attack. Despite shocks on upper surface, achieved $< 5\%$ error in lift/drag prediction with $2000\times$ speedup vs RANS CFD.

#### 3.1.3 Parametric Shape Optimization

A compelling application of neural operators is aerodynamic shape optimization, where geometry itself is a high-dimensional parameter.

*Problem:* Find airfoil shape $\mathcal{S}(\boldsymbol{\alpha})$ (parameterized by $\boldsymbol{\alpha} \in \mathbb{R}^d$, e.g., Bézier coefficients) that maximizes lift-to-drag ratio $L/D$ subject to flow equations.

*Traditional Approach:* Adjoint-based optimization requiring $\sim$100-1000 CFD evaluations (weeks of computation).

*Neural Operator Approach:* Train operator $\mathcal{G} : \boldsymbol{\alpha} \to (\mathbf{u}, p, L, D)$ on database of airfoil shapes, then use for rapid optimization.

*State-of-the-Art:* Li et al. (Li et al., 2023d) demonstrated: Training: 30,000 airfoils from UIUC database, CFD simulations at Re $= 10^6$, Architecture: Geo-FNO handling varying geometry through diffeomorphic mapping, Optimization: Evolutionary algorithm with 10,000 neural operator evaluations (30 seconds total) vs infeasible with CFD, and Result: Discovered novel airfoil design (Sekar et al., 2019)s with $L/D$ improvements of 8-12% over baseline

### 3.2 Solid Mechanics and Structural Optimization

Solid mechanics problems involve material parameter identification, design optimization under varying loading, and geometry-parameterized structural analysis—all areas where parametric neural methods are making significant contributions.

#### 3.2.1 Linear Elasticity

The linear elastic equations govern small-deformation structural response:

$$-\nabla \cdot (\mathbb{C}(\boldsymbol{\mu}) : \nabla \mathbf{u}) = \mathbf{f}, \quad \text{in } \Omega(\boldsymbol{\mu}_{\text{geom}}) \tag{54}$$

where $\mathbf{u}$ is displacement, $\mathbb{C}$ is the elasticity tensor (parameterized by Young's modulus $E$, Poisson ratio $\nu$), and $\Omega$ may be geometry-parameterized.

**Parametric Scenarios:** The problem accommodates three main types of parameters. Material parameters $\boldsymbol{\mu} = (E, \nu)$ represent Young's modulus and Poisson ratio for isotropic materials, extending to up to 21 independent components for anisotropic materials. Loading parameters capture force magnitude, spatial distribution, and directional variations. Geometry parameters encompass structural shape variations, hole positions, and thickness distributions.

**Representative Studies:**

*1. Parameter Identification with PINNs* (Haghighat et al., 2021): Haghighat et al. developed a PINN framework for identifying spatially-varying material properties from displacement measurements.

*Setup:* Unknown: Heterogeneous Young's modulus field $E(x, y)$, Observations: Displacement measurements at 100 sensor locations (1% of domain), and Prior: Smooth spatial variation, $E \in [50, 150]$ GPa

*Approach:* Treat $E(x, y)$ as an additional neural network output. Minimize combined loss:

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}} + \lambda_{\text{reg}}\|\nabla E\|^2 \tag{55}$$

where the regularization term enforces smoothness.

*Results:* Reconstructed $E(x, y)$ with $< 3\%$ error from sparse data, Simultaneously obtained displacement field at arbitrary resolution, and Uncertainty quantification via Bayesian PINN variant showed credible intervals consistent with ground truth

*2. Operator Learning for Parametric Structures* (Goswami et al., 2022): Goswami et al. applied DeepONet to bridge structures parameterized by loading patterns and geometric variations.

*Problem:* Input function: Distributed load $f(x)$ on bridge deck, Parameters: Span length $L \in [10, 30]$m, support stiffness $k$, and Output: Displacement field $\mathbf{u}(x; f, L, k)$ and stress tensor $\boldsymbol{\sigma}$

*Results:* Training: 800 FEM simulations with varying $(f, L, k)$, Accuracy: $< 5\%$ error for displacement, $< 10\%$ for stress (stress concentrations challenging), and Application: Real-time structural health monitoring—compare predicted vs measured displacements to detect damage

### 3.2.2 Nonlinear Mechanics and Plasticity

Nonlinear constitutive relations introduce severe challenges. Hyperelastic materials (rubbers, biological tissues) have strain energy functions $W(\mathbf{F}; \boldsymbol{\mu})$ dependent on material parameters $\boldsymbol{\mu}$ (e.g., Mooney-Rivlin constants). Elastoplastic materials exhibit history-dependent, irreversible deformation.

*Neural Constitutive Modeling:* Recent work (Masi et al., 2021; Thakolkaran et al., 2022) learns constitutive relations directly from data:

$$\boldsymbol{\sigma} = \mathcal{NN}_\theta(\mathbf{F}, \text{history}; \boldsymbol{\mu}) \tag{56}$$

ensuring thermodynamic consistency (e.g., positive dissipation) through architecture constraints.

*Application - Elastomeric Materials:* Thakolkaran et al. (Thakolkaran et al., 2022) trained neural operators to predict large-deformation response of elastomers across parameter space of Ogden model coefficients. Achieved $< 2\%$ error in stress-strain curves for complex multiaxial loading, enabling rapid virtual testing for material design.

### 3.2.3 Topology Optimization

Zhu et al. (Zhu et al., 2023) introduced Phase-Field DeepONet using energy-based loss functions for pattern formation, enabling fast Allen-Cahn and Cahn-Hilliard simulations. Lee et al. (Lee et al., 2025) developed FE Operator Networks for high-dimensional elasticity (d=50 parameters), achieving 60% error reduction.

Topology optimization seeks the optimal material distribution $\rho(x) \in [0, 1]$ (density) minimizing compliance subject to volume constraint:

$$\min_{\rho} c(\rho) = \int_\Omega f \cdot u(\rho)\, dx, \quad \text{s.t.} \int_\Omega \rho\, dx \leq V_{\text{max}} \tag{57}$$

where $u(\rho)$ solves elasticity with density-dependent stiffness.

*Parametric Aspect:* Loading conditions, boundary supports, and volume fractions are parameters defining different optimization scenarios.

*Neural Approach:* Chandrasekhar and Suresh (Sosnovik & Oseledets, 2019) (Chandrasekhar & Suresh, 2021) trained neural operators to map loading patterns to optimal topologies:

$$\mathcal{G} : f(x) \to \rho^*(x) \tag{58}$$

*Breakthrough:* After training on 10,000 topology optimization problems (each requiring iterative FEM), the neural operator: Generates near-optimal designs in $< 1$ second (vs hours for conventional optimization), Achieved 95-98% of optimal compliance, Handles novel loading patterns via generalization, and Enables interactive design exploration

*Impact:* Transforms topology optimization from overnight batch process to interactive design tool.

### 3.2.4 Fracture Mechanics

Crack propagation is highly parameter-sensitive: small changes in loading, material properties, or initial flaw geometry can dramatically alter fracture paths.

*Phase-Field Fracture:* The phase-field approach introduces damage variable $d(x, t) \in [0, 1]$ governed by coupled PDEs:

$$-\nabla \cdot (\sigma(u, d)) = 0, \tag{59}$$

$$d - \ell^2 \nabla^2 d = \mathcal{H}(u)/G_c, \tag{60}$$

where $\ell$ is length scale, $G_c$ is fracture toughness, and $\mathcal{H}$ is elastic energy.

*Parametric Challenge:* Parameter sweeps over $(G_c, \ell, \text{loading rate})$ require thousands of expensive phase-field simulations.

*Neural Solution:* Goswami et al. (Goswami et al., 2023) developed transfer learning PINNs: Pre-train on simple fracture geometries (Mode I cracks), Fine-tune for complex scenarios (branching, multiple cracks), and Achieves 10× speedup per parameter value while maintaining crack path accuracy

### 3.3 Heat Transfer and Conjugate Problems

Heat transfer problems naturally involve parametric variations in thermal conductivity, convection coefficients, and heat source characteristics. Conjugate heat transfer coupling fluid flow with thermal conduction adds further complexity.

#### 3.3.1 Parametric Heat Conduction

The heat equation with parameter-dependent thermal conductivity:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k(\mathbf{x}; \boldsymbol{\mu}) \nabla T) + Q(\mathbf{x}, t; \boldsymbol{\mu}) \tag{61}$$

where $k$ is thermal conductivity and $Q$ is heat source.

**Industrial Application - Electronic Cooling:** Cai et al. (Cai et al., 2021) addressed thermal management of chip packages with varying power distributions and heat sink geometries.

*Parameters:* Heat generation map $Q(x, y)$ (spatially varying chip power), Heat sink fin spacing $s \in [1, 5]$ mm, and Convection coefficient $h \in [10, 100]$ W/(m²·K)

*Neural Operator Solution:* Physics-informed DeepONet trained on 500 thermal simulations. Branch network encodes $Q(x, y)$ at sensor locations; parameters $(s, h)$ input to both networks.

*Results:* Accuracy: $<$ 2C error in maximum temperature prediction, Speedup: 5000× vs commercial thermal solver, and Design optimization: Evaluated 50,000 configurations in 10 minutes, identified optimal heat sink reducing peak temperature by 15°C

#### 3.3.2 Thermal Property Identification

*Inverse Problem:* Infer spatially-varying thermal conductivity $k(x, y)$ from limited temperature measurements—critical for characterizing novel materials or detecting defects.

*PINN Approach:* Anantha Padmanabha et al. (Anantha Padmanabha & Zabaras, 2021) developed Bayesian PINNs: Observations: Temperature at 50 locations via infrared thermography, Unknown: 2D conductivity field $k(x, y)$ with suspected discontinuities (material interfaces), and Method: PINN with total variation regularization to preserve sharp interfaces

*Achievement:* Reconstructed $k(x, y)$ including interfacial discontinuities with $< 5\%$ error, providing uncertainty maps guiding sensor placement for reduced uncertainty.

#### 3.3.3 Conjugate Heat Transfer

Conjugate problems couple fluid flow (Navier-Stokes) with heat conduction in solids, requiring matched temperature and heat flux at interfaces.

*Parametric Complexity:* Both fluid Reynolds number and solid thermal conductivity ratios affect heat transfer, creating multi-parameter coupling.

*State-of-the-Art:* Penwarden et al. (Penwarden et al., 2023) developed multi-fidelity (Penwarden et al., 2023; Howard et al., 2023; Peherstorfer et al., 2018) neural operators for conjugate heat transfer: Low fidelity: Simplified 1D thermal resistance models (cheap), High fidelity: Coupled CFD-conduction simulations (expensive), and Neural fusion: Multi-fidelity DeepONet learns correction from low to high fidelity

*Performance:* With 100 high-fidelity and 1000 low-fidelity samples, achieved $< 3\%$ error in heat transfer coefficients across parameter ranges $(\text{Re}, k_{\text{ratio}})$, enabling design optimization of heat exchangers.

### 3.4 Electromagnetics and Wave Propagation

Electromagnetic problems governed by Maxwell's equations involve material parameters (permittivity $\epsilon$, permeability $\mu$, conductivity $\sigma$) and geometric configurations, making them prime candidates for parametric neural methods.

#### 3.4.1 Parametric Maxwell's Equations

Time-harmonic Maxwell equations (Chen et al., 2020; Wiecha & Muskens, 2020) in frequency domain:

$$\nabla \times \mathbf{E} = -i\omega\mu(\mathbf{x}; \boldsymbol{\mu})\mathbf{H}, \tag{62}$$

$$\nabla \times \mathbf{H} = i\omega\epsilon(\mathbf{x}; \boldsymbol{\mu})\mathbf{E} + \sigma\mathbf{E}, \tag{63}$$

where $\omega$ is angular frequency and parameters $\boldsymbol{\mu}$ describe material distributions.

**Application - Metamaterial Design:** Chen et al. (Chen et al., 2020) used neural operators for inverse design (Lu et al., 2021b; Meng et al., 2022) of electromagnetic metamaterials.

*Problem:* Find spatial permittivity distribution $\epsilon(x, y)$ yielding desired scattering behavior (e.g., cloaking, focusing).

*Approach:* Forward operator: $\mathcal{G}_F : \epsilon(x, y) \to \mathbf{E}(x, y)$ (permittivity $\to$ field), and Inverse operator: $\mathcal{G}_I : \mathbf{E}_{\text{target}}(x, y) \to \epsilon(x, y)$ (desired field $\to$ design)

*Training:* 20,000 random permittivity configurations, full-wave simulations for each.

*Results:* Forward prediction: $< 2\%$ error in $\mathbf{E}$ field compared to finite-element solver, Inverse design: Generated metamaterial achieving 85-90% of target performance, and Design time: Seconds vs days for topology optimization

#### 3.4.2 Acoustic Wave Propagation

The acoustic wave (Jin et al., 2022b) equation in heterogeneous media:

$$\frac{1}{c(\mathbf{x}; \boldsymbol{\mu})^2}\frac{\partial^2 p}{\partial t^2} = \nabla^2 p + s(\mathbf{x}, t) \tag{64}$$

where $c$ is sound speed (material parameter) and $p$ is pressure.

*Seismic Imaging Application:* Moseley et al. (Moseley et al., 2023a) addressed full-waveform inversion for subsurface velocity model estimation.

*Parameters:* 2D velocity field $c(x, z)$ with typical dimension $\sim$10,000 (discretized).

*Challenge:* High-dimensional parameter space, expensive forward wave simulations, local minima in optimization.

*Neural Solution:* Learned neural operator $\mathcal{G} : c(x, z) \to$ seismogram enables: Gradient-based inversion $1000\times$ faster than adjoint-state methods, Better convergence by avoiding local minima through learned physics, and Uncertainty quantification via ensemble neural operators

### 3.5 Cross-Domain Insights and Application Maturity

**Common Success Patterns: Success Factors:** Neural methods achieve greatest impact in applications with well-characterized physics, where mature traditional solvers provide abundant training data for neural operator development. Multi-query scenarios such as design optimization, uncertainty quantification, and parameter studies justify the upfront neural training investment through repeated rapid evaluations. The methods particularly excel when solution manifolds exhibit smooth parameter dependence with low intrinsic dimensionality, enabling efficient learning from finite training samples.

**Remaining Barriers:** Despite these successes, significant challenges persist. High-Reynolds-number turbulent flows and chaotic dynamics pose difficulties for long-term stability in autoregressive predictions. Processes spanning more than six orders of magnitude in length or time scales present multi-scale coupling challenges that strain current architectures. Rare events including phase transitions, shock formations, and bifurcations demand specialized handling beyond standard training approaches.

Table 4: Cross-domain comparison of neural methods for parametric PDEs

| Domain | Common Parameters | Param Dim | Dominant Methods | Reported Speedup |
|---|---|---|---|---|
| Fluid Dynamics | Re, Mach, geometry | 1-10 | FNO, DeepONet, PINO | $10^3$-$10^5\times$ (Li et al., 2021; Jin et al., 2021) |
| Solid Mechanics | $E$, $\nu$, loading | 1-20 | PINN, DeepONet, GNO | $10^2$-$10^3\times$ |
| Heat Transfer | $k$, $h$, sources | 1-10 | PI-DeepONet, PINN | $10^3$-$10^4\times$ |
| Electromagnetics | $\epsilon$, $\mu$, $\omega$ | 1-5 | PINN, MaxwellNet | $10^2$-$10^3\times$ |
| Acoustics | $c$, density | 1-100 | FWI operators | $10^2$-$10^3\times$ |

**Industrial Adoption Status: Production deployment:** Weather forecasting (FourCastNet (Pathak et al., 2022; Kurth et al., 2023)), some engineering design optimization, **Pilot projects:** Digital twins, real-time control systems, medical imaging, and **Research stage:** Nuclear fusion, climate prediction, drug discovery

# 4 Theoretical Foundations and Analysis

While empirical success has driven adoption of neural methods for parametric PDEs, rigorous theoretical understanding is essential for reliability, interpretability, and principled algorithm design. This section examines the mathematical foundations underpinning physics-informed networks and neural operators, analyzing approximation capabilities, generalization behavior, and computational complexity with emphasis on parametric aspects.

## 4.1 Approximation Theory for Parametric PDEs

### 4.1.1 The Parametric Solution Manifold

Understanding the structure of solution spaces is fundamental to approximation theory. For a parametric PDE, the solution manifold

$$\mathcal{M} = \{u(\cdot;\mu) : \mu \in \mathcal{P}\} \subset \mathcal{U} \tag{65}$$

where $\mathcal{U}$ is an appropriate function space (typically Sobolev), encodes all possible solutions as parameters vary. The manifold's geometric properties—dimension, curvature, smoothness—determine approximation difficulty.

**Kolmogorov n-Width:** A classical measure of manifold complexity from reduced-order modeling theory (Binev et al., 2011; Berkooz et al., 1993; Hesthaven et al., 2016; Peherstorfer et al., 2014) is the Kolmogorov $n$-width:

$$d_n(\mathcal{M},\mathcal{U}) = \inf_{\mathcal{V}_n} \sup_{u\in\mathcal{M}} \inf_{v\in\mathcal{V}_n} \|u - v\|_{\mathcal{U}} \tag{66}$$

where the infimum is over all $n$-dimensional subspaces $\mathcal{V}_n \subset \mathcal{U}$. This quantifies how well $\mathcal{M}$ can be approximated by $n$-dimensional linear subspaces.

For elliptic PDEs with smooth, affine parameter dependence, $d_n(\mathcal{M}) \sim e^{-\alpha n}$ (exponential decay) justifies reduced basis methods (Binev et al., 2011). However, transport-dominated or hyperbolic problems may have $d_n \sim n^{-\beta}$ (polynomial decay), requiring larger $n$ for adequate approximation.

**Implications for Neural Methods:** While Kolmogorov $n$-width characterizes linear approximation, neural networks provide nonlinear approximation. The relevant question: can neural operators achieve better decay rates through nonlinear representations?

### 4.1.2 Universal Approximation Theorems for Operators

Classical universal approximation for functions $f : \mathbb{R}^n \to \mathbb{R}^m$ states neural networks can approximate continuous functions arbitrarily well. For operators, we require extensions to infinite-dimensional spaces.

**Theorem 2** (Neural Operator Universal Approximation (Kovachki et al., 2023a)). *Let $\mathcal{A}$ and $\mathcal{U}$ be compact subsets of separable Banach spaces. For any continuous operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ and $\epsilon > 0$, there exists a neural operator architecture with parameters $\theta$ such that:*

$$\sup_{a \in \mathcal{A}} \|\mathcal{G}(a) - \mathcal{G}_\theta(a)\|_{\mathcal{U}} < \epsilon. \tag{67}$$

*Proof Sketch:* The proof proceeds in three steps:

1. *Discretization:* Approximate continuous functions in $\mathcal{A}$ by finite-dimensional representations via sampling
2. *Finite approximation:* Use classical universal approximation for the finite-dimensional case
3. *Consistency:* Show approximation error vanishes as discretization refines

**Parametric Extension:** For parametric operators $\mathcal{G} : \mathcal{P} \times \mathcal{A} \to \mathcal{U}$, the theorem applies with parameters entering either as finite-dimensional inputs (concatenated to function discretizations) or through conditioning mechanisms.

**Architecture-Specific Results:**

*DeepONet:* Lu et al. (Lu et al., 2021a) proved that the branch-trunk decomposition equation 28 achieves universal approximation if: Branch network width $p \to \infty$, and Both branch and trunk networks are universal approximators

The key insight: factorization $\sum_{k=1}^{p} b_k(a)t_k(y)$ can represent any operator via appropriate choices of basis functions.

*FNO:* Li et al. (Li et al., 2021) showed Fourier neural operators approximate operators by leveraging spectral properties. For periodic problems, the Fourier series representation provides natural expressivity. The critical assumption: solutions have sufficient spectral decay (smoothness in Fourier space).

**Approximation Rates:** While existence is guaranteed, rates remain an active research area. For smooth solutions:

$$\|\mathcal{G} - \mathcal{G}_\theta\|_{\text{op}} \lesssim \frac{1}{W^{\alpha/d}} \tag{68}$$

where $W$ is network width, $d$ is input dimension, and $\alpha$ is smoothness. The curse of dimensionality ($1/W^{\alpha/d}$) persists theoretically (Han et al., 2018; Beck et al., 2020; Khoo et al., 2021), though empirically neural operators perform better than this bound suggests.

#### 4.1.3 Parametric PDE-Specific Analysis

Recent work examines approximation rates specifically for parametric PDE solutions.

**Low-Dimensional Structure:** Bhattacharya et al. (Bhattacharya et al., 2021) proved that if the solution manifold $\mathcal{M}$ has intrinsic dimension $d_{\text{eff}} \ll d$ (as measured by Kolmogorov width decay), then neural operators can achieve dimension-independent error bounds:

$$\|\mathcal{G} - \mathcal{G}_\theta\|_{\text{op}} \lesssim e^{-cW^{1/d_{\text{eff}}}} \tag{69}$$

This explains empirical success in problems where traditional theory predicts exponential complexity in $d$.

**Regularity Propagation:** For parametric elliptic PDEs, if parameters enter affinely and solutions have bounded derivatives, then neural network approximation error inherits regularity:

$$\|u(\cdot;\mu) - u_\theta(\cdot;\mu)\|_{H^k} \lesssim C(\text{network size}, \text{depth}, k) \tag{70}$$

where $H^k$ is the Sobolev space of order $k$.

### 4.2 Generalization and Parameter Space Coverage

#### 4.2.1 Training and Generalization Error Decomposition

For parametric neural methods, generalization encompasses two distinct aspects that must be considered jointly. Spatial generalization involves evaluating solutions at new spatial locations $x$ not present in the

training set, while parameter generalization requires accurate predictions for parameter values $\mu$ outside the training distribution. Both capabilities are essential for practical parametric surrogate modeling.

The total error decomposes as:

$$\mathbb{E}_{\mu,x}[|u(x;\mu) - u_\theta(x;\mu)|^2] = \underbrace{\mathbb{E}[|u - u_\theta^*|^2]}_{\text{Approximation}} + \underbrace{\mathbb{E}[|u_\theta^* - u_\theta|^2]}_{\text{Generalization}} \tag{71}$$

where $u_\theta^*$ is the best possible network in the function class.

**Parametric Complexity:** The generalization error depends on: *Parameter space coverage:* Training sample distribution in $\mathcal{P}$, *Solution smoothness:* Lipschitz constant of $\mu \mapsto u(\cdot;\mu)$, and *Network capacity:* Number of parameters relative to sample size

#### 4.2.2 Sample Complexity Bounds

**Question:** How many training samples $N$ are needed to achieve $\epsilon$-accuracy over parameter space?

De Ryck and Mishra (De Ryck & Mishra, 2022) derived sample complexity for PINNs approximating parametric elliptic PDEs:

$$N \sim \frac{d \log(1/\epsilon)}{\epsilon^2} \cdot \text{poly}(\text{network width}) \tag{72}$$

where $d$ is parameter dimension. This exhibits logarithmic dependence on accuracy but polynomial dependence on dimension—significantly better than exponential.

For neural operators, Lanthaler et al. (Lanthaler et al., 2022) proved:

$$N \sim d_{\mathcal{M}}^{1+\delta} \epsilon^{-2} \tag{73}$$

where $d_{\mathcal{M}}$ is the effective dimension of the solution manifold. When $d_{\mathcal{M}} \ll d$ (low intrinsic dimensionality), sample requirements scale favorably.

**Empirical Observations:** Practical training often uses fewer samples than theoretical bounds suggest. Possible explanations: Over-conservative bounds not accounting for problem structure, Physics constraints (in PINNs) reducing sample requirements, and Implicit regularization from optimization algorithms

#### 4.2.3 Interpolation vs. Extrapolation

A critical distinction in parametric problems: **Interpolation:** $\mu_{\text{test}} \in \text{convex hull}(\{\mu_{\text{train}}\})$, and **Extrapolation:** $\mu_{\text{test}} \notin \text{convex hull}(\{\mu_{\text{train}}\})$

**Empirical Findings:** *FNO:* Interpolation error $\sim$1-2%, extrapolation error $\sim$10-20% (Li et al. (Li et al., 2021)), *DeepONet:* Similar interpolation/extrapolation gap, slightly better extrapolation when physics-informed, and *PINN:* Poor parametric generalization—essentially no extrapolation capability without retraining

**Theoretical Understanding:** Extrapolation fundamentally requires assumptions about solution structure beyond training data. Neural operators implicitly learn these through architecture inductive biases (e.g., FNO's Fourier basis assumes periodic/smooth solutions).

#### 4.2.4 Out-of-Distribution Detection

For safety-critical applications, detecting when $\mu_{\text{test}}$ lies outside the reliable prediction region is crucial.

*Approaches:* Uncertainty quantification employs multiple complementary approaches. Bayesian neural operators compute predictive variance as an out-of-distribution indicator. Ensemble methods identify uncertain regions through high variance across ensemble members. Conformal prediction constructs prediction sets with guaranteed coverage (Staber & Da Veiga, 2024), providing distribution-free uncertainty bounds.

*Breakthrough - Conformal Prediction for PDEs:* Staber et al. (Staber & Da Veiga, 2024) developed distribution-free uncertainty quantification for neural operators:

$$\mathcal{C}(a,\mu) = \{u : \|u - \mathcal{G}_\theta(a,\mu)\|_{\mathcal{U}} \leq q_\alpha\} \tag{74}$$

where $q_\alpha$ is the $(1-\alpha)$-quantile of calibration residuals. This provides rigorous $1-\alpha$ coverage guarantees without distributional assumptions—a major advance for reliability.

## 4.3 Computational Complexity Analysis

### 4.3.1 Training Complexity

The computational cost structure differs fundamentally across methods:

**PINNs:** *Per-iteration cost:* $\mathcal{O}(N_{\text{collocation}} \cdot N_{\text{params}} \cdot \text{AD cost})$ where AD (automatic differentiation) for second derivatives is expensive, *Total training:* $\mathcal{O}(N_{\text{iter}} \cdot N_{\text{collocation}} \cdot N_{\text{params}})$, typically $N_{\text{iter}} \sim 10^4$-$10^5$, and *Bottleneck:* Computing PDE residuals via repeated differentiation

**DeepONet:** *Per-iteration:* $\mathcal{O}(N_{\text{data}} \cdot (N_{\text{branch}} + N_{\text{trunk}}))$ - standard forward pass, *Total training:* $\mathcal{O}(N_{\text{iter}} \cdot N_{\text{data}})$, with $N_{\text{iter}} \sim 10^3$-$10^4$, and *Advantage:* No automatic differentiation needed unless physics-informed

**FNO:** *Per-iteration:* $\mathcal{O}(N_{\text{data}} \cdot N_{\text{grid}} \log N_{\text{grid}})$ due to FFT, *Total training:* Similar to DeepONet but with FFT overhead, and *Scaling:* Exceptionally favorable for large spatial grids

### 4.3.2 Inference Complexity

This is where neural operators shine for parametric problems:

Table 5: Inference complexity comparison for parametric PDEs

| Method | Training (offline) | Inference (online) | Multi-Query Advantage |
|---|---|---|---|
| Traditional FEM | N/A | $\mathcal{O}(N_{\text{DOF}}^{2-3})$ per $\mu$ | None |
| Reduced Basis | $\mathcal{O}(M \cdot N_{\text{DOF}}^{2-3})$ | $\mathcal{O}(N_{\text{RB}}^3) \ll N_{\text{DOF}}^3$ | High |
| PINN | $\mathcal{O}(10^4 \cdot N_{\text{coll}})$ per $\mu$ | $\mathcal{O}(N_{\text{query}})$ | Low (retrain) |
| DeepONet | $\mathcal{O}(10^3 \cdot N_{\text{data}})$ | $\mathcal{O}(N_{\text{query}})$ | Very High |
| FNO | $\mathcal{O}(10^3 \cdot N_{\text{data}})$ | $\mathcal{O}(N \log N)$ | Very High |

**Break-Even Analysis:** Neural operators become advantageous when:

$$N_{\text{queries}} > \frac{C_{\text{training}}}{C_{\text{traditional}} - C_{\text{inference}}} \tag{75}$$

For typical parameters: DeepONet: Break-even at $\sim$10-50 queries, FNO: Break-even at $\sim$5-20 queries, and PINN: Rarely breaks even for parametric studies

### 4.3.3 Memory Requirements

**Storage:** Neural operators require storing network weights ($\sim$10-100 MB typically) vs. reduced basis methods storing basis functions ($\sim$GB for high-fidelity problems).

**Runtime Memory:** FNO: Requires full spatial field in memory ($\sim$GB for 3D problems), and DeepONet: Query-point-by-point evaluation possible, lower memory

### 4.3.4 Parallelization and Hardware Efficiency

**GPU Acceleration:** Neural operators achieve massive parallelization: Batch processing across parameters: Evaluate 100s of $\mu$ simultaneously, Spatial parallelization: All grid points computed in parallel, and Training parallelization: Data parallel across multiple GPUs

**Specialized Hardware:** Emerging neuromorphic chips and tensor processing units offer 10-100$\times$ additional speedups for inference, potentially enabling microsecond-scale PDE solving.

## 4.4 Convergence and Stability Theory

### 4.4.1 Training Convergence for PINNs

A fundamental challenge: proving that PINN training converges to PDE solutions. Recent progress:

**Neural Tangent Kernel (NTK) Analysis:** Wang et al. (Wang et al., 2022b), building on failure mode analyses (Krishnapriyan et al., 2021; 2022; Markidis, 2021; Fuks & Tchelepi, 2020), used NTK theory to analyze PINN training dynamics. For infinitely-wide networks in the NTK regime:

$$\frac{du_\theta}{dt} = -\Theta(x, x')\mathcal{L}_{\text{PDE}}(u_\theta)(x') \tag{76}$$

where $\Theta$ is the neural tangent kernel. Convergence occurs if $\Theta$ is positive definite—but this fails for: Stiff PDEs (large condition number), Multi-scale problems (eigenvalue clustering), and High-frequency solutions (spectral bias)

This explains empirically observed training failures (Krishnapriyan et al., 2021).

**Implications for Parametric Problems:** Different parameter values may have vastly different conditioning, causing training instability across parameter space.

#### 4.4.2 Operator Learning Convergence

For neural operators trained with data loss $\mathcal{L}_{\text{data}}$, convergence to the true operator $\mathcal{G}$ depends on: The total error arises from three sources that must be controlled simultaneously. Data quality determines the approximation error present in training samples. Optimization convergence affects how closely the trained network approaches the optimal parameters within the function class. Generalization capability governs uniform convergence performance across the entire parameter space.

Lanthaler et al. (Lanthaler et al., 2022) proved that for appropriate architecture choices and sufficient training data:

$$\mathbb{P}\left(\|\mathcal{G} - \mathcal{G}_\theta\|_{\text{op}} > \epsilon\right) < \delta \tag{77}$$

with sample complexity scaling as $N \sim \epsilon^{-2}\log(\delta^{-1})$—standard statistical learning theory rates.

#### 4.4.3 Long-Time Stability

For time-dependent problems, autoregressive application of neural operators can accumulate errors:

$$u_{n+1} = \mathcal{G}_\theta(u_n), \quad n = 0, 1, 2, \ldots \tag{78}$$

**Error Accumulation:** With per-step error $\epsilon$, after $T$ steps:

$$\|u_T - u_T^{\text{true}}\| \lesssim T\epsilon \cdot (1 + L)^T \tag{79}$$

where $L$ is the Lipschitz constant of $\mathcal{G}$. For $L > 0$, errors grow exponentially—catastrophic for chaotic systems.

**Mitigation Strategies:** *Markov neural operators:* Train on multi-step trajectories to learn error correction, *Correction networks:* Periodically apply high-fidelity solver corrections, and *Physics-informed training:* Enforce conservation laws to constrain error growth

*Recent Success:* Brandstetter et al. (Brandstetter et al., 2022) demonstrated stable 1000-step rollouts for turbulent flows by enforcing energy and enstrophy conservation in FNO training.

### 4.5 Theoretical Gaps and Open Questions

Despite significant progress, several fundamental questions remain:

**1. Sharp Approximation Rates:** Existing bounds are often pessimistic. Can we derive problem-specific rates that match empirical observations?

**2. Parameter-Dependent Convergence:** How does approximation quality vary across $\mu \in \mathcal{P}$? Can we identify "hard" parameter regions?

**3. Generalization Theory:** Why do neural operators generalize well with relatively few samples compared to worst-case theory?

**4. Physics-Informed Regularization:** Quantify how physics constraints improve data efficiency and generalization.

**5. Adversarial Robustness:** Can small perturbations to parameters cause large solution errors? How to certify robustness?

**Emerging Directions:** *Operator PINNs:* Combining operator learning with physics-informed training for provable convergence, *Certified methods:* Formal verification of neural PDE solvers, and *Adaptive approximation:* Theoretical guidance for architecture selection based on PDE properties

# 5 Advanced Topics and Emerging Directions

This section explores cutting-edge developments addressing the most challenging aspects of parametric PDEs: high-dimensional parameter spaces, rigorous uncertainty quantification, rapid adaptation to new parameter regimes, and hybrid approaches combining neural methods with traditional solvers.

## 5.1 High-Dimensional Parameter Spaces

When parameter dimension $d$ exceeds 50-100, even neural operators face challenges. We examine strategies for tractable high-dimensional parametric PDE solving.

### 5.1.1 The Curse of Dimensionality Revisited

**Sampling Complexity:** To uniformly cover a $d$-dimensional unit hypercube with spacing $h$ requires $(1/h)^d$ samples—exponential in $d$. For $d = 100$ and $h = 0.1$, this yields $10^{100}$ samples (intractable).

**Volume Concentration:** In high dimensions, almost all volume concentrates near boundaries and corners. Random sampling becomes inefficient as typical samples lie far from any training point.

**Neural Network Perspective:** While neural networks mitigate dimensionality to some extent through hierarchical representations, approximation error bounds still exhibit polynomial or exponential dependence on $d$ in worst-case analysis.

### 5.1.2 Active Subspaces and Dimension Reduction

Many high-dimensional parameter spaces have low effective dimensionality. Active subspace methods (Constantine, 2015) identify important parameter directions.

**Mathematical Framework:** Define the active subspace matrix:

$$C = \mathbb{E}_{\mu\sim\pi}\left[\nabla_\mu Q(\mu)\nabla_\mu Q(\mu)^T\right] \tag{80}$$

where $Q(\mu)$ is a quantity of interest. Eigendecomposition $C = V\Lambda V^T$ reveals: Large eigenvalues: Active directions (significant QoI variation), and Small eigenvalues: Inactive directions (negligible QoI variation)

**Parametric PDE Application:** Project parameters onto active subspace:

$$\mu = \bar{\mu} + V_{\text{active}}\xi, \quad \xi \in \mathbb{R}^{d_{\text{eff}}} \text{ with } d_{\text{eff}} \ll d \tag{81}$$

then train neural operators on the reduced space $\xi$.

**Case Study:** Hu et al. (Hu et al., 2024) addressed a subsurface flow (Tang et al., 2020; Fuks & Tchelepi, 2020) problem with $d = 100,000$ uncertain permeability parameters (each grid cell). Active subspace analysis revealed $d_{\text{eff}} = 5$ active directions explaining 95% of pressure variance. After reduction: Trained DeepONet on 5D active subspace, Achieved $< 3\%$ error in pressure predictions, and Total computation: 12 hours on single GPU (vs. years for full-space sampling)

### 5.1.3 Sparse and Low-Rank Representations

**Tensor Decompositions:** High-dimensional parameter dependence often admits low-rank structure:

$$u(x,t;\mu_1,\ldots,\mu_d) \approx \sum_{k=1}^{r} u_k(x,t)\prod_{j=1}^{d}\phi_{jk}(\mu_j) \tag{82}$$

(canonical polyadic decomposition). Neural operators can learn this structure implicitly.

**Hierarchical Representations:** Group parameters hierarchically. For example, in materials with microstructure, parameters might be: Macro-scale: $\mu_{\text{macro}} \in \mathbb{R}^{10}$ (global properties), Meso-scale: $\mu_{\text{meso}} \in \mathbb{R}^{100}$ (grain structures), and Micro-scale: $\mu_{\text{micro}} \in \mathbb{R}^{10000}$ (defects)

Neural operators with hierarchical architectures (U-Net-style) naturally capture multi-scale parameter effects without explicit reduction.

#### 5.1.4 Latent Variable Models

For extremely high-dimensional parameters, learn a latent encoding:

$$\mu \in \mathbb{R}^d \to z \in \mathbb{R}^{d_z} \to u(\cdot; z) \tag{83}$$

where $d_z \ll d$.

**Architecture:** Encoder: Enc : $\mathbb{R}^d \to \mathbb{R}^{d_z}$, and Neural operator: $\mathcal{G}_\theta : \mathbb{R}^{d_z} \to \mathcal{U}$

**Training:** Joint optimization over encoder and operator using available PDE solutions.

**L-DeepONet:** Kontolati et al. (Kontolati et al., 2024) developed latent-space DeepONet for high-dimensional parametric PDEs: Application: Structural dynamics with $d = 10,000$ (discretized forcing field) (Rao et al., 2021; Rezaei et al., 2022; Samaniego et al., 2020), Latent dimension: $d_z = 50$ (learned via variational autoencoder), and Results: 1-2 orders of magnitude speedup vs. standard DeepONet, $< 5\%$ accuracy loss

#### 5.1.5 Sensitivity Analysis for Parameter Prioritization

When $d$ is large but not all parameters are equally important, sensitivity analysis guides resource allocation.

**Sobol Indices:** Variance-based sensitivity:

$$S_i = \frac{\text{Var}_{\mu_i}[\mathbb{E}_{\mu_{\sim i}}[Q(\mu)|\mu_i]]}{\text{Var}[Q(\mu)]} \tag{84}$$

quantifies fraction of output variance due to parameter $\mu_i$.

**Morris Screening:** Compute elementary effects:

$$EE_i = \frac{Q(\mu + \Delta e_i) - Q(\mu)}{\Delta} \tag{85}$$

to identify influential parameters with few evaluations.

**Neural Operator Integration:** Train neural operators only on high-sensitivity parameters, treating others as fixed or marginalized. Taneja et al. (Taneja et al., 2023) demonstrated $10\times$ training data reduction for 50-dimensional aerodynamics problems by focusing on top 10 sensitive parameters.

### 5.2 Uncertainty Quantification

Rigorous UQ is essential for high-stakes applications. We examine Bayesian and conformal approaches for neural parametric PDE solvers.

#### 5.2.1 Bayesian Physics-Informed Neural Networks

B-PINNs (Yang et al., 2021) place priors over network weights:

$$p(\theta|\text{data}) \propto p(\text{data}|\theta)p(\theta)p_{\text{PDE}}(\theta) \tag{86}$$

where $p_{\text{PDE}}(\theta)$ encodes physics constraints as likelihood term.

**Inference:** Variational inference or Hamiltonian Monte Carlo produces posterior samples $\{\theta^{(s)}\}_{s=1}^S$. Predictions include uncertainty:

$$p(u(x;\mu)|\text{data}) \approx \frac{1}{S}\sum_{s=1}^{S} \delta(u - u_{\theta^{(s)}}(x;\mu)) \tag{87}$$

**Advantages:** Uncertainty in parameter identification: $p(\mu|\text{data})$, Prediction intervals: $[Q_{0.025}(u), Q_{0.975}(u)]$, and Model selection: Compare PDE formulations via marginal likelihood

**Computational Cost:** Sampling from posterior requires training multiple networks—expensive but parallelizable.

**Application - Cardiovascular Flows:** Arzani et al. (Arzani et al., 2021), along with related cardiovascular modeling studies (Kissas et al., 2020; Sahli Costabal et al., 2020; Yazdani et al., 2020), used B-PINNs to infer patient-specific blood viscosity and vessel compliance from sparse Doppler ultrasound data. Posterior distributions quantified parameter uncertainty, enabling risk assessment (e.g., probability of flow reversal).

#### 5.2.2 Bayesian Neural Operators

Extending Bayesian ideas to operator learning:

$$p(\mathcal{G}_\theta|\{(a_i, u_i)\}) \propto \prod_{i=1}^{N} p(u_i|\mathcal{G}_\theta(a_i))p(\theta) \tag{88}$$

**Challenges:** Operator space is higher-dimensional than typical PINN weight spaces, making posterior inference more expensive.

**Efficient Approximations:** *Laplace approximation:* Gaussian approximation around MAP estimate, *Concrete dropout:* Approximate variational inference via dropout (Gal & Ghahramani, 2016), and *Ensemble methods:* Train multiple operators with different initializations

**Empirical Study:** Psaros et al. (Psaros et al., 2023) compared UQ methods for turbulent flows: B-DeepONet: Most rigorous but 10× computational cost, Deep ensemble (10 networks): Good coverage, 5× cost, and MC Dropout: Fastest but underestimates uncertainty

#### 5.2.3 Conformal Prediction for Distribution-Free UQ

Recent breakthrough: conformal prediction provides coverage guarantees without distributional assumptions (Staber & Da Veiga, 2024).

**Framework:** Given calibration set $\{(\mu_i, u_i)\}_{i=1}^{N_{\text{cal}}}$ and miscoverage level $\alpha$:

1. Compute conformity scores: $R_i = \|u_i - \mathcal{G}_\theta(\mu_i)\|$
2. Find quantile: $\hat{q} = \text{Quantile}_{1-\alpha}(\{R_i\})$
3. Prediction set: $\mathcal{C}(\mu) = \{u : \|u - \mathcal{G}_\theta(\mu)\| \leq \hat{q}\}$

**Guarantee:** $\mathbb{P}(u_{\text{true}} \in \mathcal{C}(\mu)) \geq 1 - \alpha$ for any data distribution (finite-sample guarantee).

**Advantages:** No training overhead—uses any pre-trained neural operator, Valid for any distribution (no assumptions), and Finite-sample guarantees (not asymptotic)

**Challenge:** Prediction sets can be large (conservative) if neural operator has heteroscedastic errors.

**Adaptive Conformal:** Construct parameter-dependent quantiles:

$$\hat{q}(\mu) = \text{Quantile}_{1-\alpha}(\{R_i : \mu_i \text{ near } \mu\}) \tag{89}$$

using local calibration. This tightens sets while maintaining coverage.

#### 5.2.4 Uncertainty Propagation Through Parametric PDEs

Given parameter distribution $\pi(\mu)$, compute statistics of $Q(u(\mu))$:

$$\mathbb{E}[Q] = \int_{\mathcal{P}} Q(u(\mu))\pi(\mu)d\mu \tag{90}$$

**Monte Carlo with Neural Operators:**

1. Sample $\{\mu^{(s)}\}_{s=1}^{S} \sim \pi(\mu)$
2. Evaluate: $Q^{(s)} = Q(\mathcal{G}_\theta(\mu^{(s)}))$ (fast with neural operator)
3. Estimate: $\hat{\mathbb{E}}[Q] = \frac{1}{S}\sum_s Q^{(s)}$

**Computational Advantage:** $S = 10^6$ samples feasible in minutes with FNO, enabling accurate tail probability estimation ($p < 10^{-4}$) infeasible with traditional solvers.

**Multi-Level Extensions:** Combine neural operators at multiple fidelities: Level 0: Coarse physics-based model (cheap), Level 1: Neural operator trained on moderate data, and Level 2: High-fidelity validation samples (expensive, few)

Multi-fidelity Monte Carlo achieves optimal bias-variance tradeoff (Peherstorfer et al., 2018).

### 5.3 Meta-Learning and Rapid Adaptation

Meta-learning (Finn et al., 2017; Yin et al., 2022; Huang et al., 2022) enables neural methods to quickly adapt to new parameter regimes or PDE types with minimal retraining—crucial when parameter distributions shift or new physics are encountered.

#### 5.3.1 Model-Agnostic Meta-Learning (MAML) for PDEs

MAML (Finn et al., 2017) learns initialization $\theta_0$ such that few gradient steps on new tasks yield good performance.

**Algorithm:**

1. **Meta-training:** Sample task (parameter value) $\tau_i \sim p(\mathcal{T})$, Adapt: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}(\theta)$ (inner loop), and Meta-update: $\theta \leftarrow \theta - \beta \nabla_\theta \sum_i \mathcal{L}_{\tau_i}(\theta_i')$ (outer loop)
2. **Meta-test:** For new task $\tau_{\text{new}}$, fine-tune from $\theta_0$ with few samples

**PDE Application:** Huang et al. (Huang et al., 2022) applied MAML to parametric Burgers equation: Meta-train on $\nu \in [0.01, 0.05]$, Meta-test on $\nu \in [0.06, 0.10]$ (outside training distribution), Result: 5-10 gradient steps achieve $< 5\%$ error vs. 1000+ steps without meta-learning, and $100\times$ sample efficiency for new parameter values

#### 5.3.2 Transfer Learning Across Parameter Regimes

**Scenario:** Train on "easy" parameter region, transfer to "hard" region.

*Example:* Fluid flows: Pre-train: Laminar flows (Re $< 1000$)—smooth solutions, easy to learn, and Transfer: Turbulent flows (Re $> 5000$)—fine structures, challenging

**Strategy:**

1. Train neural operator on large dataset at easy parameters
2. Freeze low-level feature extractors
3. Fine-tune high-level layers on small dataset at hard parameters

**Results:** Desai et al. (Desai et al., 2021) demonstrated: Transfer from Re $= 100$ to Re $= 1000$: 80% reduction in training iterations, Maintains 95% of full-training accuracy, and Critical: transferred representations capture flow physics invariant to Re

#### 5.3.3 Meta-Auto-Decoder for Parametric PDEs

Yin et al. (Yin et al., 2022) developed Meta-Auto-Decoder combining meta-learning with latent representations:

**Architecture:**

$$u(\cdot; \mu) = \text{Decoder}_\theta(z(\mu), \cdot) \tag{91}$$

where $z(\mu)$ is a learnable latent code for parameter $\mu$.

**Meta-Learning:** Learn decoder $\theta$ such that for new $\mu$, optimizing $z$ with few PDE evaluations yields accurate $u$.

**Application - Airfoil Shapes:** 500 airfoil geometries for meta-training. For novel airfoil: Optimize latent code $z$ using 10 CFD samples, Decode to full flow field, and Achieves $< 3\%$ error vs. 100+ samples needed without meta-learning

#### 5.3.4 Continual Learning for Evolving Parameter Distributions

In dynamic environments, parameter distributions shift over time. Continual learning prevents catastrophic forgetting while adapting to new data.

**Techniques:** *Elastic weight consolidation:* Penalize changes to important weights, *Rehearsal:* Interleave old and new parameter samples during training, and *Dynamic architectures:* Expand network capacity for new parameter regions

**Application - Climate Modeling:** As climate changes, parameter distributions (e.g., atmospheric $CO_2$, temperature patterns) evolve. Continual learning enables neural operators to update without forgetting historical patterns (Chattopadhyay et al., 2023).

### 5.4 Hybrid Methods and Multi-Fidelity Approaches

Combining neural operators with traditional solvers leverages strengths of both: physics fidelity from solvers, efficiency from neural methods.

#### 5.4.1 Neural Operators as Preconditioners

In iterative PDE solvers (conjugate gradient, GMRES), preconditioning accelerates convergence:

$$Ax = b \to M^{-1}Ax = M^{-1}b \tag{92}$$

where $M \approx A$ is preconditioner.

**Neural Preconditioner:** Train neural operator $\mathcal{N}_\theta$ to approximate $A^{-1}$:

1. Use $x^{(0)} = \mathcal{N}_\theta(b)$ as initial guess
2. Apply few iterations of traditional solver for refinement

**Advantages:** Reduces iteration count by 5-10×, Preserves exact solution (solver provides correction), and Neural operator trained offline on representative problems

**Results:** Margenberg et al. (Margenberg et al., 2023) achieved 8× speedup for parametric elliptic PDEs by using FNO as preconditioner, with final solution matching traditional solver machine precision.

#### 5.4.2 Hybrid Physics-ML Models (PINO Framework)

PINO (Li et al., 2024b) combines data-driven learning with physics-informed losses at multiple resolutions:

**Multi-Resolution Loss:**

$$\mathcal{L}_{\text{PINO}} = \underbrace{\mathcal{L}_{\text{data}}^{\text{coarse}}}_{\text{cheap data}} + \lambda \underbrace{\mathcal{L}_{\text{PDE}}^{\text{fine}}}_{\text{physics}} + \gamma \underbrace{\mathcal{L}_{\text{data}}^{\text{fine}}}_{\text{few samples}} \tag{93}$$

**Strategy:**

1. Train on abundant coarse-resolution data (cheap simulations)
2. Enforce physics at fine resolution via PDE residuals
3. Fine-tune with scarce high-fidelity data

**Results:** For Navier-Stokes turbulence: Coarse data: 1000 DNS at $64 \times 64$ resolution Fine data: 50 DNS at $256 \times 256$ PINO accuracy: $< 2\%$ error at fine resolution Pure data-driven: $> 8\%$ error (insufficient fine data) Pure physics-informed: $> 5\%$ error (coarse data underutilized)

PINO achieves best of both worlds: data efficiency from physics, accuracy from multi-fidelity (Penwarden et al., 2023; Howard et al., 2023; Peherstorfer et al., 2018) data.

#### 5.4.3 FEM-Neural Operator Coupling

Directly couple finite element methods with neural operators for multi-physics or multi-scale problems.

**Example - Fluid-Structure Interaction:** Fluid: Neural operator for Navier-Stokes (fast, parametric), Structure: FEM for elasticity (accurate, geometry-adaptive), and Interface: Iterate between operators, enforcing displacement/traction continuity

**Advantage:** Each subdomain uses optimal method. Neural operator accelerates parametric fluid solve while FEM handles complex structural geometry.

**Implementation:** Koric and Abueidda (Koric & Abueidda, 2023) demonstrated: $20\times$ speedup vs. fully-coupled FEM, $< 5\%$ error in interface stresses (critical quantity), and Enables real-time digital twins for manufacturing processes (Zobeiry & Humfeld, 2021; Koric & Abueidda, 2023)

#### 5.4.4 Residual Learning and Error Correction

Train neural networks to predict solver error:

$$u_{\text{accurate}} = u_{\text{coarse}} + \mathcal{E}_\theta(\mu) \tag{94}$$

where $u_{\text{coarse}}$ is cheap approximate solution and $\mathcal{E}_\theta$ learns the correction.

**Training:** Generate: $(u_{\text{coarse}}, u_{\text{fine}})$ pairs at various $\mu$, and Learn: $\mathcal{E}_\theta : \mu \to u_{\text{fine}} - u_{\text{coarse}}$

**Inference:**

1. Compute coarse solution (fast traditional solver)
2. Predict correction (fast neural operator)
3. Sum for accurate result

**Benefit:** Combines speed of coarse solver with accuracy of neural correction.Failsafe: if neural operator fails, coarse solution is still physical.

**Application - Combustion:** Chen et al. (Chen et al., 2023) used residual learning for parametric combustion chemistry: Coarse: Simplified chemistry model ($10\times$ faster), Correction: Neural operator trained on detailed chemistry, and Result: Detailed chemistry accuracy at simplified cost

### 5.5 Foundation Models for PDEs

Inspired by large language models, foundation models for PDEs aim to create general-purpose solvers via pre-training on diverse PDE families.

#### 5.5.1 The Foundation Model Paradigm

**Concept:** Pre-train a single massive neural operator on: Multiple PDE types (elliptic, parabolic, hyperbolic), Various parameter ranges, Different domains and boundary conditions, and Multi-physics couplings

Then fine-tune for specific applications with minimal data.

#### 5.5.2 Representative Approaches

**1. Poseidon:** Hao et al. (Hao et al., 2023c) pre-trained on 15 PDE families: Architecture: Transformer-based operator with 100M parameters, Pre-training: 1M PDE solutions across equations, parameters, domains, Fine-tuning: 10-100 samples for new PDEs, and Performance: Zero-shot transfer to new PDEs with $< 10\%$ error; fine-tuning achieves $< 3\%$

**2. DPOT (Deep Physical Operator Transformer):** Another foundation model focusing on physical priors: Embeds conservation laws in architecture, Trained on synthetic + real physics data, and Generalizes across spatial dimensions (1D, 2D, 3D)

**3. MOFS (Multi-Operator Foundation Solver):** Specializes in operator composition: Learns elemental operators (advection, diffusion, reactions), Composes for complex PDEs, and Analogous to "building blocks" in language models

#### 5.5.3 Challenges and Open Questions

**1. Data Requirements:** Pre-training requires massive computational resources (months on GPU clusters). Cost-benefit tradeoffs unclear.

**2. Transfer Limitations:** Performance degrades significantly for PDEs far from training distribution. How to characterize "distance" in PDE space?

**3. Interpretability:** Foundation models are black boxes. Can we understand what physics they learn?

**4. Reliability:** Critical applications demand guarantees. How to certify foundation model predictions?

**Future Outlook:** Foundation models represent the frontier of neural PDE solving. Success would enable "ChatGPT for physics"—natural language specification of problems, automatic solution generation, democratizing computational science.

## 6 Software Tools and Benchmarks

The maturation of neural methods for parametric PDEs is reflected in emerging software ecosystems and standardized benchmarks. This section surveys practical tools and evaluation frameworks essential for researchers and practitioners.

### 6.1 Open-Source Software Frameworks

#### 6.1.1 DeepXDE (Lu et al., 2021a)

**Developer:** Brown University (Karniadakis group)

**Focus:** Physics-informed neural networks and DeepONet

**Repository:** `github.com/lululxvi/deepxde` (∼2,500 stars)

**Key Features:** Unified API for PINNs, DeepONet, and variants Moreover, Multiple backend support (TensorFlow, PyTorch, JAX, PaddlePaddle) Additionally, Built-in parametric PDE examples (Burgers, Navier-Stokes, elasticity) Furthermore, Automatic differentiation for arbitrary-order derivatives Also, Adaptive sampling strategies In addition, Multi-GPU training support

**Parametric Capabilities:** Easy parameter specification: `add_parameter(``mu", [0.01, 0.1])`, Branch-trunk architecture for operator learning, and Physics-informed constraints for parametric domains

**Strengths:** Extensive documentation, active community, beginner-friendly **Limitations:** Focus on PINNs limits scalability to very large problems

**Typical Use Case:**

```
import deepxde as dde

# Define parametric PDE
def pde(x, u, mu):
    du_t = dde.grad.jacobian(u, x, i=0, j=1)
    du_xx = dde.grad.hessian(u, x, i=0, j=0)
    return du_t - mu * du_xx  # mu-parameterized diffusion

# Create geometry and problem
geom = dde.geometry.Interval(0, 1)
timedomain = dde.geometry.TimeDomain(0, 1)
geomtime = dde.geometry.GeometryXTime(geom, timedomain)

# Specify parameter range
param_space = dde.ParameterSpace([0.01, 0.1], "mu")

# Train...
```

#### 6.1.2 NVIDIA MODULUS

**Developer:** NVIDIA

**Focus:** Industrial-scale physics-informed ML

**Repository:**

`github.com/NVIDIA/modulus` (∼600 stars)

**Key Features:** Production-ready for engineering workflows Moreover, Optimized for NVIDIA GPUs (multi-GPU, mixed precision) Additionally, FNO, PINN, and hybrid implementations Furthermore, CAD geometry integration via signed distance functions Also, Pretrained models for common physics In addition, Distributed training with Horovod

**Parametric Capabilities:** Parametric geometry via SDF modulation, Multi-parameter configuration files, and Uncertainty quantification modules

**Strengths:** Performance optimization, industry partnerships, CAD integration **Limitations:** NVIDIA hardware dependency, steeper learning curve

**Target Users:** Engineering firms, automotive industry, energy sector

#### 6.1.3 NEURALOPERATOR

**Developer:** Caltech/NVIDIA (Anandkumar group) **Focus:** Neural operator architectures

**Repository:**

`github.com/neuraloperator/neuraloperator` (∼1,400 stars)

**Key Features:** Reference implementations of FNO, GNO, GINO, Geo-FNO Moreover, Modular design for architecture experimentation Additionally, Integration with PDEBench datasets Furthermore, Multi-resolution training utilities Also, Tensorized Fourier layers for efficiency

**Parametric Capabilities:** Parameter conditioning at multiple layers, Seamless handling of functional + parametric inputs, and Zero-shot super-resolution evaluations

**Strengths:** Cutting-edge architectures, research-oriented, excellent for prototyping **Limitations:** Less documentation than DeepXDE (Lu et al., 2021a), fewer built-in examples

**Example - Parametric FNO:**

```
from neuraloperator import FNO2d

model = FNO2d(
    modes1=12, modes2=12,  # Fourier modes
    width=64,  # Channel width
    in_channels=3,  # input + parameters
    out_channels=1
)

# Input: [batch, x, y, channels]
# channels = [initial_condition, param1, param2]
u = model(input_tensor)
```

#### 6.1.4 COMPARATIVE OVERVIEW

### 6.2 BENCHMARK DATASETS AND EVALUATION PROTOCOLS

Standardized benchmarks are critical for reproducible research and fair method comparison. We survey major datasets for parametric PDEs.

#### 6.2.1 PDEBENCH

**Reference:** Takamoto et al. (Takamoto et al., 2022) **Repository:** `github.com/pdebench/PDEBench`

Table 6: Comparison of major software frameworks for parametric PDEs

| Framework | Primary Methods | Ease of Use | Performance | Best For |
|---|---|---|---|---|
| DeepXDE (Lu et al., 2021a) | PINN, DeepONet | High | Medium | Research, education, prototyping |
| NVIDIA Modulus | FNO, PINN, hybrid | Medium | Very High | Production, large-scale engineering |
| Neuraloperator | FNO family | Medium | High | Neural operator research, benchmarking |
| PyDEns | PINN | High | Low-Medium | Teaching, simple problems |

**Scope:** Comprehensive suite covering diverse PDE families with parametric variations.

**Included Problems:**

1. *1D Advection:* Wave speeds $c \in [0.5, 2.0]$
2. *1D Burgers:* Viscosity $\nu \in [0.001, 0.1]$, various initial conditions
3. *2D Navier-Stokes:* Reynolds numbers Re $\in [100, 10000]$, forcing variations
4. *2D Shallow Water:* Bathymetry parameters, Coriolis force
5. *2D Darcy Flow:* Heterogeneous permeability fields (parametric coefficients)
6. *3D Compressible Euler:* Mach numbers, specific heat ratios

**Data Format:** HDF5 files with spatiotemporal grids, Metadata: parameter values, domain specifications, timestamps, and Multiple resolutions: coarse ($64^2$) to fine ($512^2$)

**Evaluation Metrics:** Relative $L^2$ error: $\frac{\|u-u_{\text{pred}}\|_{L^2}}{\|u\|_{L^2}}$ Moreover, Maximum pointwise error Additionally, Parameter-averaged error: $\mathbb{E}_\mu[\text{error}(\mu)]$ Furthermore, Out-of-distribution error (extrapolation) Also, Inference time per parameter configuration

**Leaderboard:** Public leaderboard tracks state-of-the-art across methods and problems.

**Impact:** PDEBench enables apples-to-apples comparisons, accelerating research progress.

### 6.2.2 Domain-Specific Benchmarks

**1. Cylinder Flow Dataset (Fluid Dynamics)** Parameters: Re $\in [40, 500]$, cylinder diameter $d \in [0.5, 2.0]$, 1000 CFD simulations (OpenFOAM), Ground truth: velocity, pressure, vorticity fields, and Challenge: Capture von Kármán vortex street across parameters

**2. Elasticity Benchmark (Solid Mechanics)** Problems: cantilever beams, plates with holes, L-shaped domains, Parameters: Young's modulus $E$, Poisson ratio $\nu$, loading distribution, 500 FEM solutions per geometry class, and Evaluation: stress concentration factor prediction accuracy

**3. Airfoil Database (Aerodynamics)** UIUC airfoil database: 1550 airfoil shapes, Parameters: Shape coefficients + Re, Mach, angle of attack, Lift, drag, moment coefficient targets, and Enables data-driven aerodynamic optimization

### 6.2.3 Standardized Evaluation Protocol

To ensure reproducibility, we recommend the following protocol for parametric PDE studies:

**Data Splitting:** Training: 70% of parameter space (randomly sampled), Validation: 15% (for hyperparameter tuning), Test: 15% (held out, reported results), and OOD test: Additional samples outside training parameter range

**Reported Metrics:**

1. Accuracy: Mean and standard deviation of relative $L^2$ error over test parameters
2. Interpolation vs. extrapolation errors separately
3. Worst-case error: $\max_{\mu \in \mathcal{P}_{\text{test}}} \text{error}(\mu)$
4. Inference time: Mean and 95th percentile
5. Training cost: GPU-hours

**Ablation Studies:** Vary training data size: $[10, 50, 100, 500, 1000]$ samples, Vary parameter dimension (if applicable), Compare with/without physics constraints, and Sensitivity to hyperparameters (learning rate, architecture)

**Code Release:** Provide runnable code and trained model checkpoints for reproducibility.

# 7 Challenges and Future Directions

While neural methods for parametric PDEs have achieved remarkable successes, fundamental challenges remain. This section critically examines limitations and identifies promising research directions.

## 7.1 Current Limitations

### 7.1.1 Theoretical Gaps

**1. Incomplete Convergence Guarantees**

*Problem:* For PINNs, no universal convergence theorem exists. Training can fail unpredictably, especially for: Stiff PDEs with multiple timescales, High Reynolds number flows (thin boundary layers), and Problems with sharp gradients or discontinuities

*Evidence:* Krishnapriyan et al. (Krishnapriyan et al., 2021) systematically documented failure modes: Diffusion-dominated vs. advection-dominated regime switching causes training collapse, Multi-scale problems exhibit oscillatory loss with no convergence, and Spectral bias prevents learning high-frequency components

*Research Need:* Develop theory predicting when PINNs converge, and design architectures with provable guarantees.

**2. Loose Generalization Bounds**

*Problem:* Existing sample complexity bounds are overly pessimistic, often predicting exponential data requirements when empirically modest datasets suffice.

*Gap:* Theory vs. practice mismatch suggests we don't understand why neural operators generalize so well. Possible explanations: Low intrinsic dimensionality of solution manifolds (not captured in worst-case analysis), Implicit regularization from stochastic gradient descent, and Architecture inductive biases (Fourier basis, graph symmetries) not reflected in theory

*Research Need:* Problem-dependent generalization bounds incorporating PDE structure.

**3. Parameter-Dependent Convergence Rates**

*Problem:* Approximation quality varies wildly across parameter space. Some parameter regions converge rapidly, others barely learn.

*Example:* For Navier-Stokes, neural operators excel at Re $\sim$ 1000 but struggle at Re $>$ 5000 (turbulent transition).

*Research Need:* Characterize "easy" vs. "hard" parameter regions a priori, enabling targeted data collection or method selection.

### 7.1.2 Practical Challenges

**1. Training Instability and Hyperparameter Sensitivity**

*Problem:* PINNs require careful loss balancing:

$$\mathcal{L} = \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}} \tag{95}$$

where optimal $\{\lambda_i\}$ vary by problem and aren't known a priori.

*Symptom:* One loss term dominates, others ignored; solution satisfies some constraints but violates others.

*Current Approaches:* Learning rate annealing, gradient balancing (Wang et al., 2021b), uncertainty weighting (Kendall et al., 2018)—but no universal solution.

**2. Data Requirements for Neural Operators**

*Problem:* Pure data-driven neural operators (FNO, DeepONet without physics) require 1000s of training samples—expensive for high-fidelity simulations.

*Cost Analysis:* If each training simulation costs 1 GPU-hour: 1000 samples = 1000 GPU-hours (∼$1000-5000 on cloud), and Additional development time: weeks to months

*Mitigation:* Physics-informed training (PINO, PI-DeepONet) reduces data needs 5-10×, but tuning physics loss weight is non-trivial.

**3. Out-of-Distribution Generalization**

*Problem:* Neural methods interpolate well but extrapolate poorly. For parameters outside training distribution, errors increase dramatically.

*Example:* Trained on Re $\in [100, 1000]$, tested on Re $= 1500$: Interpolation (Re $= 550$): 2% error, and Extrapolation (Re $= 1500$): 25% error

*Consequence:* Limits applicability to scenarios with unexpected parameter values (safety-critical systems, climate tipping points).

*Research Direction:* Integrate physical bounds and monotonicity constraints to guide extrapolation.

**4. Computational Cost of Training**

*Problem:* Training neural operators from scratch is expensive: Typical training: 10-100 GPU-hours, and Foundation models: 1000s of GPU-hours

*Comparison:* For a single parameter value, traditional solver might cost 1-10 GPU-hours. Break-even requires many queries to justify.

*Emerging Solution:* Transfer learning and foundation models amortize costs across problems.

### 7.1.3 Domain-Specific Challenges

**1. Turbulence and Chaos**

*Problem:* Turbulent and chaotic systems are fundamentally challenging: Sensitive dependence on initial conditions, Multi-scale energy cascades (6+ orders of magnitude), and Long-time instability in autoregressive rollout

*Current Status:* FNO achieves ∼50 timestep stable rollout for 2D turbulence, but 3D or longer horizons remain elusive.

*Approach:* Enforce conservation laws (energy, enstrophy) as hard constraints in architecture.

**2. Multi-Physics Coupling**

*Problem:* Coupled phenomena (fluid-structure interaction, chemically reacting flows) involve disparate timescales and physics.

*Challenge:* Training single neural operator for all physics vs. modular operators for each physics with coupling—unclear which is better.

**3. Topological Changes**

*Problem:* Some parametric scenarios involve topology changes: Phase transitions (solidification, melting), Crack initiation and propagation, and Bubble coalescence in multiphase flows

*Difficulty:* Standard neural operators assume fixed topology. Representing topology changes in learned representations is open problem.

### 7.2 Future Research Directions

We identify five high-priority research directions with transformative potential.

#### 7.2.1 Direction 1: Theoretical Foundations

**Motivation:** Rigorous theory is essential for safety-critical applications and guides algorithm design.

**Key Questions:**

1. *Approximation:* Derive parameter-dependent approximation rates. When does low-rank structure in solution manifold emerge?
2. *Generalization:* Prove sample complexity bounds incorporating PDE structure (smoothness, conservation laws, symmetries).
3. *Optimization:* Characterize loss landscape for physics-informed methods. When do local minima trap training?
4. *Extrapolation:* Develop theory distinguishing safe vs. unsafe parameter extrapolation.

**Promising Approaches:** *Operator approximation theory:* Extend functional analysis tools (Kolmogorov width, manifold learning) to neural operators., *NTK analysis:* Leverage neural tangent kernel framework to analyze PINN convergence for specific PDE classes., and *Statistical learning theory:* Adapt PAC learning and Rademacher complexity to infinite-dimensional settings.

**Expected Impact:** Enable principled method selection ("use FNO for smooth solutions, GNO for irregular geometry"), predictable training outcomes, and certified predictions.

#### 7.2.2 Direction 2: Geometry-Parameterized Problems

**Motivation:** Many high-impact applications involve parametric geometry (shape optimization, patient-specific medicine, digital twins).

**Current State:** Progress via Geo-FNO, GNO, DIMON—but handling large geometric variations and topological changes remains difficult.

**Open Problems:**

1. *Representation:* How to parameterize geometry families (aircraft shapes, biological organs) in low-dimensional latent space?
2. *Generalization:* Neural operators trained on one geometry class (e.g., car shapes) rarely transfer to different class (e.g., airplanes).
3. *Topology:* No satisfactory method for topology-changing scenarios.

**Promising Approaches:** *Implicit representations:* Use level sets, signed distance functions, or neural implicit representations (NeRF-style) as geometry parameterization., *Diffeomorphic mapping:* Build on DIMON's success—extend to larger geometry variations, automate mapping network training., *Equivariant operators:* Design neural operators with geometric equivariance (rotation, scaling invariance)., and *Topology-adaptive architectures:* Dynamic graph networks that add/remove nodes/edges to handle topology changes.

**Target Applications:** Automotive design: Optimize aerodynamics across full design space, Medical imaging: Patient-specific simulations from scan to diagnosis in minutes, and Zhu et al. (Zhu et al., 2023) introduced Phase-Field DeepONet using energy-based loss functions for pattern formation, enabling fast Allen-Cahn and Cahn-Hilliard simulations. Lee et al. (Lee et al., 2025) developed FE Operator Networks for high-dimensional elasticity (d=50 parameters), achieving 60% error reduction.

Topology optimization: Real-time structural design exploration

#### 7.2.3 Direction 3: Foundation Models and Transfer Learning

**Vision:** Pre-train massive neural operators on diverse PDE families, enabling few-shot adaptation to new problems—"GPT for physics."

**Current Status:** Early prototypes (Poseidon, DPOT) show promise but face challenges: Training cost: Thousands of GPU-hours, Generalization limits: Performance degrades for PDEs far from training distribution, and Interpretability: Unclear what physics is learned

**Research Opportunities:**

1. *Architecture design:* What inductive biases enable broad PDE learning? (Conservation laws, causality, multi-scale hierarchies?)
2. *Pre-training objectives:* Beyond supervised learning—can self-supervised or contrastive learning on synthetic data improve transfer?
3. *Modular composition:* Learn "building block" operators (advection, diffusion, reaction) that compose for complex PDEs.
4. *Continual learning:* Update foundation models as new physics data becomes available without catastrophic forgetting.

**Practical Considerations:** *Data curation:* Build large-scale PDE simulation databases (millions of samples across equations, parameters, domains)., *Community models:* Open-source pre-trained models like ImageNet for computer vision or BERT for NLP., and *Fine-tuning protocols:* Standardized pipelines for adapting foundation models to specific applications.

**Potential Impact:** Democratize computational science—non-experts could solve complex PDEs via natural language specification and foundation model inference.

#### 7.2.4 Direction 4: Uncertainty Quantification and Reliability

**Motivation:** Safety-critical applications (aerospace, medical, nuclear) demand rigorous UQ and failure detection.

**Current Gaps:** Most neural methods provide point predictions without uncertainties, Bayesian approaches are expensive, and No standard for "how much uncertainty is acceptable"

**Key Challenges:**

1. *Epistemic vs. aleatoric:* Distinguish uncertainty from limited training data (epistemic) vs. inherent randomness (aleatoric).
2. *Calibration:* Neural networks are often overconfident—predicted probabilities don't match true frequencies.
3. *Out-of-distribution detection:* Identify when parameters lie outside reliable prediction region.
4. *Worst-case guarantees:* Provide bounds on maximum possible error.

**Promising Approaches:** *Conformal prediction:* Distribution-free coverage guarantees (breakthrough: Staber et al. 2024)—extend to sequential predictions and multi-fidelity (Penwarden et al., 2023; Howard et al., 2023; Peherstorfer et al., 2018) settings., *Bayesian neural operators:* Make tractable via variational inference, Laplace approximation, or ensemble distillation., *Certified robustness:* Borrow techniques from adversarial ML—prove that small parameter perturbations cause bounded solution changes., and *Physics-based validation:* Check predictions against conservation laws, positivity constraints, maximum principles as sanity checks.

**Standardization Needs:** Benchmark UQ methods on common parametric PDE suite, Define industry-specific reliability requirements, and Develop best practices for reporting uncertainties

#### 7.2.5 Direction 5: Integration with Scientific Discovery

**Vision:** Use neural operators not just as fast solvers but as tools for discovering new physics, materials, and designs.

**Paradigms:**

*1. Inverse Design:* Given desired solution properties, find parameters:

$$\mu^* = \arg\min_{\mu} \text{Loss}(\mathcal{G}_\theta(\mu), u_{\text{target}}) \tag{96}$$

*Applications:* Metamaterial design: Target electromagnetic response → material structure, Drug design: Desired binding affinity → molecular geometry, and Climate intervention: Target temperature reduction → intervention parameters

*Challenge:* Inverse problems are often ill-posed—multiple parameters yield similar solutions. Regularization and prior knowledge essential.

*2. Active Learning and Optimal Experimental Design:*

Neural operators enable cheap "what-if" queries, guiding expensive experiments:

1. Train neural operator on initial data
2. Use acquisition function (e.g., expected improvement) to select next parameter
3. Perform experiment at selected parameter
4. Update neural operator, repeat

*Success Story:* Lookman et al. (Lookman et al., 2019) used this loop for materials discovery, reducing experiments by $10\times$ to find optimal composition.

*3. Equation Discovery:*

Learn governing equations from data using neural operators as differentiable simulators (Raissi, 2018; Brunton et al., 2016; Zhang & Lin, 2018; Geneva & Zabaras, 2020; Chen et al., 2018): Sparse regression (SINDy) (Brunton et al., 2016) with neural-operator-generated training data, Neural differential equations (Chen et al., 2018) with interpretable coefficients, and Symbolic regression guided by neural operator gradients (Zhang & Lin, 2018)

**Research Directions:** *Interpretable operators:* Design architectures whose learned representations map to physical concepts (energy, momentum, vorticity)., *Hypothesis testing:* Use neural operators to rapidly evaluate thousands of hypotheses about parameter effects., and *Multi-objective optimization:* Navigate trade-offs (e.g., maximize lift, minimize drag, constraint stress) in parametric design spaces.

### 7.3 Community Recommendations

To accelerate progress, we propose the following community-level initiatives:

**1. Standardized Benchmarks**

Establish comprehensive benchmark suite (Takamoto et al., 2022) covering: Diverse PDE types (elliptic, parabolic, hyperbolic, mixed), Parameter dimensions from 1 to 100+, Multiple application domains, and Varying difficulty levels

Maintain public leaderboard with reproducible baselines.

**2. Open-Source Ecosystem**

Develop unified API across frameworks (inspired by scikit-learn), Create model zoo of pre-trained neural operators, Build dataset repository with standardized formats, and Establish code review and quality standards

**3. Interdisciplinary Collaboration**

*Math + ML:* Develop theory bridging functional analysis and deep learning, *Domain science + ML:* Co-design methods with physicists, engineers, clinicians, and *HPC + ML:* Optimize implementations for exascale computing

Host workshops at major conferences (ICML, NeurIPS, SIAM) bringing together these communities.

**4. Reproducibility Standards**

Require for publication: Public code repositories with installation instructions Moreover, Trained model checkpoints Additionally, Detailed hyperparameter specifications Furthermore, Computational cost reporting (GPU-hours) Also, Random seed documentation

**5. Education and Outreach**

Develop curricula blending numerical PDEs and ML, Create online tutorials and Jupyter notebooks, Organize summer schools and bootcamps, and Write accessible reviews for domain scientists

**Long-Term Vision:** Establish neural methods for parametric PDEs as a mature, reliable subdiscipline with: Theoretical foundations comparable to traditional numerical analysis, Industrial adoption for routine engineering tasks, Democratized access enabling non-experts to solve complex physics problems, and Proven track record in safety-critical applications

This requires sustained effort from researchers, funding agencies, and industry partners—but the potential impact on science and engineering justifies the investment.

### 7.4 Context and Related Surveys

This survey builds upon and extends several complementary surveys and foundational works in the field. Comprehensive overviews of physics-informed machine learning have been provided by Cuomo et al. (Cuomo et al., 2022), Azizzadenesheli et al. (Azizzadenesheli et al., 2024), and Karniadakis et al., establishing the broader context for physics-informed neural networks. Theoretical foundations in deep learning for PDEs were laid by Han et al. (Han et al., 2018), E and Yu (E & Yu, 2018), Sirignano and Spiliopoulos (Sirignano & Spiliopoulos, 2018), and Berg and Nyström (Berg & Nyström, 2018), demonstrating feasibility of neural approximations for high-dimensional problems.

Reduced-order modeling foundations relevant to parametric PDEs are comprehensively covered by Hesthaven et al. (Hesthaven et al., 2016), Berkooz et al. (Berkooz et al., 1993), and Peherstorfer et al. (Peherstorfer et al., 2014), providing mathematical context for solution manifold structure. Methodological comparisons and benchmarking efforts (Lu et al., 2022; Geneva & Zabaras, 2022; Gao et al., 2022; Fuks & Tchelepi, 2020) have critically evaluated trade-offs between different neural approaches.

Domain-specific advances complement our parametric focus: geophysical applications (Bihlo & Popovych, 2022), quantum mechanics (Hermann et al., 2020), chemical kinetics (Ji et al., 2021), manufacturing (Zobeiry & Humfeld, 2021; Cai et al., 2022), nanophotonics (Wiecha & Muskens, 2020), and financial mathematics (Ruf & Wang, 2020) demonstrate breadth of applicability. Meta-learning and adaptive approaches (Huang et al., 2022; 2023; Hao et al., 2023a; Geneva & Zabaras, 2020; Goswami et al., 2020) address rapid adaptation challenges critical for parametric problems.

Advanced architectural innovations including graph-based methods (Hao et al., 2023b; Gao et al., 2022), Hamiltonian-preserving networks (Greydanus et al., 2019), wavelet-based operators (Gupta et al., 2021), factorized representations (Tran et al., 2023), and geometric approaches (Baqué et al., 2015) continue expanding the toolkit. Multi-fidelity strategies (Howard et al., 2023; Penwarden et al., 2023) and enhanced conservation law enforcement (Jagtap et al., 2020) address data efficiency and physical consistency.

Our contribution distinguishes itself through systematic focus on parametric aspects—how methods handle parameter variations, generalize across parameter spaces, and enable multi-query applications—providing unified perspective on an increasingly fragmented literature.

## 8 Conclusion

This comprehensive survey has examined the rapid evolution of neural methods for solving parametric partial differential equations, with emphasis on physics-informed neural networks and neural operators. The field has matured remarkably since the seminal works of Raissi et al. (Raissi et al., 2019) and Lu et al. (Lu et al., 2021a), transitioning from proof-of-concept demonstrations to practical engineering tools achieving $10^3$-$10^5\times$ computational speedups while maintaining or exceeding traditional solver accuracy.

## 8.1 Key Achievements and Insights

### 8.1.1 Algorithmic Breakthroughs

The introduction of operator learning paradigms represents a fundamental shift in computational PDE solving. Unlike traditional methods that solve for specific parameter values, neural operators learn mappings from entire parameter spaces to solution spaces. This enables:

**1. Amortized Computation:** Once trained, neural operators provide near-instantaneous predictions across parameter space. DeepONet and FNO achieve inference times of milliseconds compared to hours for traditional solvers, enabling applications previously considered computationally intractable: Real-time design optimization exploring $10^6$ configurations, Interactive digital twins for manufacturing and healthcare, Monte Carlo uncertainty quantification with $10^6$ samples, and Parametric sensitivity analysis for high-dimensional systems

**2. Zero-Shot Generalization:** FNO's ability to evaluate at resolutions not seen during training breaks the traditional resolution-computation tradeoff. This mesh-free property enables adaptive refinement without retraining and natural handling of multi-resolution data.

**3. Geometric Flexibility:** Recent advances (Geo-FNO, DIMON, GNO) have overcome the fixed-geometry limitation that plagued early neural methods. The ability to handle parametric shapes opens transformative applications in patient-specific medicine, aerospace design, and topology optimization.

**4. Hybrid Methods:** The PINO framework and physics-informed variants demonstrate that combining data-driven learning with physics constraints provides the best of both worlds: data efficiency from physics knowledge and accuracy from empirical observations. This hybrid paradigm achieves superior performance with 5-10$\times$ fewer training samples than pure data-driven approaches.

### 8.1.2 Theoretical Understanding

While practical applications have progressed rapidly, theoretical foundations have also advanced significantly:

**Universal Approximation:** Rigorous proofs establish that neural operators can approximate any continuous operator between function spaces (Kovachki et al., 2023b; Lu et al., 2021a), providing mathematical legitimacy to the approach.

**Generalization Theory:** Sample complexity bounds, while still conservative, explain why neural operators require fewer training samples than naive dimensional analysis suggests—the key insight being that PDE solution manifolds often have low intrinsic dimensionality despite high ambient dimension.

**Failure Mode Analysis:** Systematic studies (Krishnapriyan et al., 2021; Wang et al., 2022b) have identified when and why physics-informed methods struggle (spectral bias, stiff equations, multi-scale problems), guiding algorithm development and honest assessment of applicability.

**Uncertainty Quantification:** The advent of conformal prediction for PDEs (Staber & Da Veiga, 2024) provides distribution-free coverage guarantees, addressing a critical need for reliable uncertainty estimates in safety-critical applications.

### 8.1.3 Cross-Domain Impact

Neural methods have demonstrated utility across the full spectrum of computational science:

**Fluid Dynamics:** From weather forecasting (FourCastNet (Pathak et al., 2022; Kurth et al., 2023) achieving competitive accuracy with traditional NWP at 1000$\times$ speedup) to turbulence modeling and aerodynamic optimization—Reynolds number parameterization enables rapid design iteration.

**Solid Mechanics:** Zhu et al. (Zhu et al., 2023) introduced Phase-Field DeepONet using energy-based loss functions for pattern formation, enabling fast Allen-Cahn and Cahn-Hilliard simulations. Lee et al. (Lee et al., 2025) developed FE Operator Networks for high-dimensional elasticity (d=50 parameters), achieving 60% error reduction.

Topology optimization transformed from overnight batch process to interactive design tool. Material parameter identification from sparse measurements enables structural health monitoring and quality control.

**Heat Transfer:** Thermal management of electronics, laser processing optimization, and inverse identification of thermal properties all benefit from parametric neural solvers' ability to explore design spaces efficiently.

**Electromagnetics:** Metamaterial inverse design (Lu et al., 2021b; Meng et al., 2022), antenna optimization, and Maxwell equation solving for varying material properties demonstrate the technology's versatility across physics domains.

**Multi-Physics:** Fluid-structure interaction, conjugate heat transfer, and chemically reacting flows showcase neural operators' potential for coupled problems where traditional partitioned approaches face stability challenges.

### 8.2 Remaining Challenges

Despite impressive progress, significant obstacles remain before neural methods can fully replace traditional PDE solvers in production environments:

#### 8.2.1 Reliability and Trust

**Predictable Convergence:** Unlike traditional solvers with well-understood convergence theory, neural methods can fail unpredictably. PINN training sometimes simply doesn't converge, with no a priori indication of failure. This unpredictability hinders adoption in risk-averse industries (aerospace, nuclear, medical).

**Verification and Validation:** Traditional solvers undergo decades of V&V before production use. Neural methods lack standardized validation protocols. How do we know a neural operator hasn't learned spurious correlations that fail catastrophically outside the training distribution?

**Error Estimation:** While traditional solvers provide rigorous error estimates and convergence rates, most neural methods offer only empirical error assessments on test sets. Developing reliable a posteriori error indicators remains an open challenge.

**Certification:** Safety-critical applications require formal guarantees that predictions satisfy physical constraints (e.g., positivity, conservation laws, causality). Current approaches enforce constraints approximately during training, but certified architectures with hard guarantees are needed.

#### 8.2.2 Computational Barriers

**Training Cost:** While inference is fast, training remains expensive—typically 10-100 GPU-hours for moderate problems, thousands for foundation models. For single-query problems, traditional solvers are more efficient.

**Data Requirements:** Pure data-driven operators require 1000s of training samples. Even physics-informed variants need hundreds. For problems where high-fidelity simulations cost hours, generating training data becomes a bottleneck.

**Hyperparameter Sensitivity:** Neural methods have many architectural and training choices (network depth/width, learning rate, loss weighting, initialization). Optimal settings vary by problem with limited theory guiding selection. This contrasts with traditional methods' mature understanding.

**Scalability to 3D:** Most success stories are 1D/2D problems. Three-dimensional problems with complex geometries remain computationally challenging—memory requirements for 3D grids and training time scale unfavorably.

#### 8.2.3 Fundamental Limitations

**Chaotic Systems:** Long-time prediction of chaotic dynamics (turbulence, weather beyond 2-week horizon) remains elusive. Errors accumulate exponentially in autoregressive rollout, and neural operators haven't solved this fundamental challenge.

**Discontinuities:** Shocks, contact discontinuities, and phase transitions violate smoothness assumptions underlying neural approximation. While progress has been made (conservative formulations, shock-capturing layers), performance lags traditional shock-capturing schemes.

**Topological Changes:** Crack propagation, phase transitions, and free-surface flows involve topology changes that current architectures handle poorly. Representing non-homeomorphic solution spaces in neural networks is conceptually challenging.

**Extreme Multi-Scale:** Problems spanning $> 6$ orders of magnitude in length/time scales (e.g., turbulent combustion, multi-phase flows) challenge both traditional and neural methods, but neural approaches lack the decades of specialized techniques (adaptive mesh refinement, implicit-explicit schemes) developed for traditional solvers.

### 8.3 Practical Recommendations

For researchers and practitioners considering neural methods for parametric PDEs, we offer the following guidance:

#### 8.3.1 For Researchers

**1. Choose Problems Strategically:** Neural methods excel when: Many parameter queries are needed (multi-query scenarios), Traditional solvers are expensive (hours per solve), Geometry varies parametrically (shape optimization, patient-specific), and Real-time inference is required (control, digital twins)

Avoid applying neural methods when traditional solvers are already fast ($<$1 minute), single queries suffice, or data generation is prohibitively expensive.

**2. Invest in Data Quality:** Neural operators are only as good as their training data. High-quality simulations covering parameter space well are essential. Multi-fidelity approaches can reduce costs but require careful calibration.

**3. Leverage Physics:** Pure data-driven learning requires massive datasets. Physics-informed training, conservation law enforcement, and symmetry-aware architectures dramatically improve data efficiency. The best results combine data and physics.

**4. Validate Thoroughly:** Test on held-out parameters, extrapolation scenarios, and physically extreme cases. Compare against traditional solvers on challenging benchmarks. Quantify uncertainty and failure modes honestly.

**5. Contribute to Community Resources:** Share trained models, datasets, and code. Participate in benchmark development. Document failures as well as successes to advance collective understanding.

#### 8.3.2 For Practitioners

**1. Start with Established Methods:** For production applications, prioritize mature frameworks (DeepXDE (Lu et al., 2021a), NVIDIA Modulus) with community support and track records. Avoid cutting-edge methods until well-validated.

**2. Hybrid Approaches First:** Rather than replacing traditional solvers entirely, integrate neural operators as components (preconditioners, surrogate models, parameter screening) within established workflows. This provides fallback to traditional methods if neural components fail.

**3. Invest in Training Infrastructure:** Successful deployment requires GPU clusters, data management systems, and MLOps pipelines. Treat neural operator training as analogous to wind tunnel testing or high-fidelity simulation campaigns—significant upfront investment enabling downstream efficiency.

**4. Develop Internal Expertise:** Neural methods require different skill sets than traditional simulation. Teams need expertise in machine learning, optimization, and software engineering alongside domain knowledge. Training programs and collaborations with academic researchers can build capabilities.

**5. Regulatory and Validation:** For regulated industries, engage with regulatory bodies early to establish acceptable validation protocols. Document training procedures, architecture choices, and performance meticulously. Plan for regular retraining as new data becomes available.

### 8.4 Concluding Remarks

Neural methods for parametric PDEs represent a genuine paradigm shift in computational science, not merely incremental improvement. The ability to solve entire parametric families of PDEs in seconds after training opens applications impossible with traditional approaches: real-time optimization, interactive design, massive ensemble simulations, and digital twins operating at decision-making timescales.

However, this survey has also highlighted that neural methods are not universal replacements for traditional solvers. They excel in specific niches—multi-query scenarios with moderate accuracy requirements—but struggle with single-query problems, long-time chaotic predictions, and applications demanding certified guarantees. The future likely involves coexistence and integration: neural operators accelerating parametric exploration while traditional solvers provide verification, handle extreme cases, and ensure physical fidelity.

The next decade will determine whether neural methods transition from research curiosity to production workhorse. Success requires sustained effort across multiple fronts: theoretical rigor to understand capabilities and limitations, algorithmic innovation to address current shortcomings, software engineering to build reliable tools, benchmark development to enable fair comparison, and application demonstrations to build trust.

For the computational science community, this is a pivotal moment. Neural methods offer the potential to solve problems at scales previously impossible, accelerating scientific discovery and engineering innovation. Realizing this potential demands collaboration across disciplines—mathematicians providing theory, machine learning researchers developing algorithms, domain scientists identifying applications, and software engineers building infrastructure.

The parametric PDE solving problem, which motivated reduced-order modeling for decades, has found powerful new tools in physics-informed neural networks and neural operators. As these methods mature, they will not replace the rich tradition of numerical analysis and scientific computing but rather complement and extend it, enabling simulations at speeds and scales that open new frontiers.

## References

G. Anantha Padmanabha and N. Zabaras. Solving inverse problems using conditional invertible neural networks. *Journal of Computational Physics*, 433:110194, 2021. doi: 10.1016/j.jcp.2021.110194.

A. Arzani, J.-X. Wang, and R. M. D'Souza. Uncovering near-wall blood flow from sparse data with physics-informed neural networks. *Physics of Fluids*, 33(7):071905, 2021. doi: 10.1063/5.0055600.

K. Azizzadenesheli, N. Kovachki, Z. Li, M. Liu-Schiaffini, J. Kossaifi, and A. Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6:320–328, 2024. doi: 10.1038/s42254-024-00712-5.

P. Baqué, E. Remelli, F. Fleuret, and P. Fua. Geodesic convolutional neural networks on riemannian manifolds. pp. 832–840, 2015. doi: 10.1109/ICCVW.2015.112.

C. Beck, F. Hornung, M. Hutzenthaler, A. Jentzen, and T. Kruse. Overcoming the curse of dimensionality in the numerical approximation of allen-cahn partial differential equations via truncated full-history recursive multilevel picard approximations. *Journal of Numerical Mathematics*, 28(4):197–222, 2020. doi: 10.1515/jnma-2019-0074.

J. Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018. doi: 10.1016/j.neucom.2018.06.056.

G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–575, 1993. doi: 10.1146/annurev.fl.25.010193.002543.

K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart. Model reduction and neural networks for parametric pdes. *The SMAI Journal of Computational Mathematics*, 7:121–157, 2021. doi: 10.5802/smai-jcm.74.

A. Bihlo and R. O. Popovych. Physics-informed neural networks for the shallow-water equations on the sphere. *Journal of Computational Physics*, 456:111024, 2022. doi: 10.1016/j.jcp.2022.111024.

P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM Journal on Mathematical Analysis*, 43(3):1457–1472, 2011. doi: 10.1137/100795772.

J. Brandstetter, D. Worrall, and M. Welling. Message passing neural pde solvers. 2022.

S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15): 3932–3937, 2016. doi: 10.1073/pnas.1517384113.

Ricardo Buitrago, Tanya Marwah, Albert Gu, and Andrej Risteski. On the benefits of memory for modeling time-dependent pdes. In *International Conference on Learning Representations (ICLR)*, 2025.

H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004. doi: 10.1017/S0962492904000182.

S. Cai, Z. Wang, L. Lu, T. A. Zaki, and G. E. Karniadakis. Deepm&mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021. doi: 10.1016/j.jcp.2021.110296.

S. Cai, Z. Wang, C. Chryssostomidis, and G. E. Karniadakis. Heat transfer prediction with unknown thermal boundary conditions using physics-informed neural networks. *ASME Journal of Heat Transfer*, 144: 011301, 2022. doi: 10.1115/1.4052555.

S. Cao. Choose a transformer: Fourier or galerkin. 34:24924–24940, 2021.

S. Cao, C. Long, and J. Zhou. Galerkin transformer: a one-shot experiment with transformer in cfd. *arXiv preprint arXiv:2305.13265*, 2023.

Wenbo Cao and Weiwei Zhang. An analysis and solution of ill-conditioning in physics-informed neural networks. *Journal of Computational Physics*, 520:113494, 2025. doi: 10.1016/j.jcp.2024.113494.

Elsa Cardoso-Bihlo and Alex Bihlo. Exactly conservative physics-informed neural networks and deep operator networks for dynamical systems. *Neural Networks*, 181:106826, 2025.

A. Chandrasekhar and K. Suresh. Tounn: Topology optimization using neural networks. *Structural and Multidisciplinary Optimization*, 63:1135–1149, 2021. doi: 10.1007/s00158-020-02748-4.

A. Chattopadhyay, M. Mustafa, P. Hassanzadeh, and K. Kashinath. Towards physics-inspired data-driven weather forecasting: integrating data assimilation with a deep spatial-transformer-based u-net in a case study with era5. *Geoscientific Model Development*, 16:3627–3641, 2023. doi: 10.5194/gmd-16-3627-2023.

R. Chen, Z. Wang, L. Lu, and G. E. Karniadakis. Residual-based attention and connection to information bottleneck theory in pinns. *arXiv preprint arXiv:2307.00166*, 2023.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. 31: 6571–6583, 2018.

T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6 (4):911–917, 1995. doi: 10.1109/72.392253.

Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, 28(8):11618–11633, 2020. doi: 10.1364/OE.384875.

Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable physics-informed neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

P. G. Constantine. Active subspaces: Emerging ideas for dimension reduction in parameter studies. 2015. doi: 10.1137/1.9781611973860.

S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific machine learning through physics-informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92:88, 2022. doi: 10.1007/s10915-022-01939-z.

A. Daw, J. Bu, S. Wang, P. Perdikaris, and A. Karpatne. Rethinking the importance of sampling in physics-informed neural networks. *arXiv preprint arXiv:2207.02338*, 2022.

T. De Ryck and S. Mishra. Error estimates for physics-informed neural networks approximating the navier-stokes equations. *Advances in Computational Mathematics*, 48(6):79, 2022. doi: 10.1007/s10444-022-09989-2.

Tim De Ryck, Florent Bonnet, Siddhartha Mishra, and Emmanuel de Bézenac. An operator preconditioning perspective on training in physics-informed machine learning. In *International Conference on Learning Representations (ICLR)*, 2024.

S. Desai, M. Mattheakis, S. Roberts, and P. Protopapas. One-shot transfer learning for differential equations. *arXiv preprint arXiv:2111.07965*, 2021.

S. Dong and Z. Li. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387: 114129, 2022. doi: 10.1016/j.cma.2021.114129.

W. E and B. Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6:1–12, 2018. doi: 10.1007/s40304-018-0127-z.

Dapeng Feng, Chaopeng Shen, et al. Sensitivity-constrained fourier neural operators for forward and inverse problems in parametric differential equations. *arXiv preprint arXiv:2505.08740*, 2025a.

Dapeng Feng, Chaopeng Shen, et al. One-shot learning for solution operators of partial differential equations. *Nature Communications*, 16, September 2025b. doi: 10.1038/s41467-025-63076-z.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pp. 1126–1135, 2017.

O. Fuks and H. A. Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1):19–37, 2020. doi: 10.1615/JMachLearnModelComput.2020033905.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. pp. 1050–1059, 2016.

H. Gao, L. Sun, and J.-X. Wang. Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids*, 33:073603, 2021. doi: 10.1063/5.0054312.

H. Gao, M. J. Zahr, and J.-X. Wang. Physics-informed graph neural galerkin networks: A unified framework for solving pde governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390:114502, 2022. doi: 10.1016/j.cma.2021.114502.

N. Geneva and N. Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020. doi: 10.1016/j.jcp.2019.109056.

N. Geneva and N. Zabaras. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022. doi: 10.1016/j.neunet.2021.11.022.

M. B. Giles. Multilevel monte carlo methods. *Acta Numerica*, 24:259–328, 2015. doi: 10.1017/S096249291500001X.

S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106: 102447, 2020. doi: 10.1016/j.tafmec.2019.102447.

S. Goswami, M. Yin, Y. Yu, and G. E. Karniadakis. A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391: 114587, 2022. doi: 10.1016/j.cma.2022.114587.

S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106: 102447, 2023. doi: 10.1016/j.tafmec.2019.102447.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. 32, 2019.

G. Gupta and J. Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *NeurIPS Workshop on Machine Learning and the Physical Sciences*, 2022. arXiv:2209.15616.

G. Gupta, X. Xiao, and P. Bogdan. Multiwavelet-based operator learning for differential equations. 34: 24048–24062, 2021.

E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021. doi: 10.1016/j.cma.2021.113741.

J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018. doi: 10.1073/pnas.1718942115.

Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, and J. Zhu. Physics-informed machine learning: A survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2023a.

Z. Hao, Z. Wang, H. Su, C. Ying, Y. Dong, S. Liu, Z. Cheng, J. Song, and J. Zhu. Gnot: A general neural operator transformer for operator learning. 2023b.

Z. Hao, C. Ying, S. Liu, H. Su, J. Zhu, and Y. Zhang. Physics-informed foundation models. *arXiv preprint arXiv:2310.02994*, 2023c.

Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, and Jun Zhu. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

J. Hermann, Z. Schätzle, and F. Noé. Deep-neural-network solution of the electronic schrödinger equation. *Nature Chemistry*, 12:891–897, 2020. doi: 10.1038/s41557-020-0544-y.

J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. SpringerBriefs in Mathematics. Springer, 2016. doi: 10.1007/978-3-319-22470-1.

P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. Turbulence, coherent structures, dynamical systems and symmetry. *Cambridge Monographs on Mechanics*, 1996.

A. A. Howard, M. Perego, G. E. Karniadakis, and P. Stinis. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Physical Review Research*, 5:023210, 2023. doi: 10.1103/PhysRevResearch.5.023210.

Z. Hu, K. Shukla, G. E. Karniadakis, and K. Kawaguchi. Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks*, 176:106369, 2024. doi: 10.1016/j.neunet.2024.106369.

X. Huang, T. Alkhalifah, C. Song, and W. Lü. Meta-auto-decoder: a meta-learning based reduced order model for solving parametric partial differential equations. *arXiv preprint arXiv:2111.08823*, 2022.

Y. Huang, X. Guo, W. Deng, and Y. Zhang. A data-driven physics-informed neural network for predicting the viscosity of nanofluids. *AIP Advances*, 13:025206, 2023. doi: 10.1063/5.0132846.

Youngsik Hwang and Dong-Young Lim. Dual cone gradient descent for training physics-informed neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Bruno Jacob, Amanda A. Howard, and Panos Stinis. Spikans: Separable physics-informed kolmogorov–arnold networks. *arXiv preprint arXiv:2411.06286*, 2024.

A. D. Jagtap and G. E. Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020. doi: 10.4208/cicp.OA-2020-0164.

A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020. doi: 10.1016/j.cma.2020.113028.

W. Ji, W. Qiu, Z. Shi, S. Pan, and S. Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125(36):8098–8106, 2021. doi: 10.1021/acs.jpca.1c05102.

P. Jin, S. Meng, and L. Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 2022a. arXiv:2202.06137.

P. Jin, S. Meng, and L. Lu. Learning poroelasticity from observation: leveraging graph neural networks and meta-learning. *Computer Methods in Applied Mechanics and Engineering*, 401:115675, 2022b. doi: 10.1016/j.cma.2022.115675.

X. Jin, S. Cai, H. Li, and G. E. Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021. doi: 10.1016/j.jcp.2020.109951.

Marimuthu Kalimuthu, David Holzmüller, and Mathias Niepert. Loglo-fno: Efficient learning of local and global features in fourier neural operators. In *International Conference on Learning Representations (ICLR)*, 2025.

A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7482–7491, 2018. doi: 10.1109/CVPR.2018.00781.

Y. Khoo, J. Lu, and L. Ying. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435, 2021. doi: 10.1017/S0956792520000182.

G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623, 2020. doi: 10.1016/j.cma.2019.112623.

Qiancheng Kong et al. Reducing frequency bias of fourier neural operators in 3d seismic wavefield modeling. *Seismological Research Letters*, 2025.

K. Kontolati, S. Goswami, G. E. Karniadakis, and M. D. Shields. Learning in latent spaces improves the predictive accuracy of deep neural operators. *Nature Communications*, 15:7770, 2024. doi: 10.1038/s41467-024-52320-y.

S. Koric and D. W. Abueidda. Data-driven and physics-informed deep learning operators for solution of heat conduction equation with parametric heat source. *International Journal of Heat and Mass Transfer*, 203:123809, 2023. doi: 10.1016/j.ijheatmasstransfer.2022.123809.

N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces. *Journal of Machine Learning Research*, 24(89):1–97, 2023a. arXiv preprint arXiv:2108.08481, 2021.

N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023b.

A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 26548–26560, 2021.

A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. doi: 10.1016/j.jcp.2021.110768.

T. Kurth, S. Subramanian, P. Harrington, J. Pathak, M. Mardani, D. Hall, A. Miele, K. Kashinath, and A. Anandkumar. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. *arXiv preprint arXiv:2208.05419*, 2023.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. 30:6402–6413, 2017.

S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for deeponets: A deep learning framework in infinite dimensions. *Transactions on Machine Learning Research*, 2022. arXiv:2102.09618.

Lise Le Boudec, Emmanuel de Bézenac, Louis Serrano, Ramon Daniel Regueiro-Espino, Yuan Yin, and Patrick Gallinari. Learning a neural solver for parametric pdes to enhance physics-informed methods. In *International Conference on Learning Representations (ICLR)*, 2025.

Jae Yong Lee, Seungchan Ko, and Youngjoon Hong. Finite element operator network for solving elliptic-type parametric pdes. *SIAM Journal on Scientific Computing*, 2025.

Haolin Li, Yuyang Miao, Zahra Sharif Khodaei, and M. H. Aliabadi. An architectural analysis of deeponet and a general extension of the physics-informed deeponet model on solving nonlinear parametric partial differential equations. *Neurocomputing*, 611:128675, 2025a.

Shanda Li, Shinjae Yoo, and Yiming Yang. Maximum update parametrization and zero-shot hyperparameter transfer for fourier neural operators. In *International Conference on Machine Learning (ICML)*, 2025b.

Tianyu Li, Shufan Zou, Xinghua Chang, Laiping Zhang, and Xiaogang Deng. Predicting unsteady incompressible fluid dynamics with finite volume informed neural network. *Physics of Fluids*, 2024a.

Tianyu Li, Yiye Zou, Shufan Zou, Xinghua Chang, Laiping Zhang, and Xiaogang Deng. Learning to solve pdes with finite volume-informed neural networks in a data-free approach. *Journal of Computational Physics*, 2025c.

Xuhui Li, Yue Wang, Chi Zhang, et al. D-fno: A decomposed fourier neural operator for large-scale parametric partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 436:117732, 2025d. doi: 10.1016/j.cma.2025.117732.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Graph kernel network for partial differential equations. *ICLR Workshop on Integration of Deep Neural Models and Differential Equations*, 2020a. arXiv:2003.03485.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, and A. Anandkumar. Multipole graph neural operator for parametric partial differential equations. 33:6755–6766, 2020b.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar. Geometry-informed neural operator for large-scale 3d pdes. 36, 2023a.

Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24:1–26, 2023b. Also appeared as Geo-FNO.

Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar. Multipole graph neural operator for parametric partial differential equations. 36, 2023c.

Z. Li, R. Ranade, D. Bianchi, F. Mueller, A. Anandkumar, and B. Liu. Physics-informed neural operator for computational fluid dynamics. *arXiv preprint arXiv:2301.01178*, 2023d.

Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1 (1):1–27, 2024b. arXiv:2111.03794.

Yunfeng Liao, Jiawen Guan, and Xiucheng Li. Curvature-aware graph attention for pdes on manifolds. 2025.

Mario Lino, Tobias Pfaff, and Nils Thuerey. Learning distributions of complex fluid simulations with diffusion graph networks. In *International Conference on Learning Representations (ICLR)*, 2025.

B. List, L. Chen, and N. Thuerey. Learned turbulence modelling with differentiable fluid solvers. *arXiv preprint arXiv:2202.06988*, 2023.

Xiaoyi Liu and Hao Tang. Difffno: Diffusion fourier neural operator. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Shane E. Loeffler, Zan Ahmad, Syed Yusuf Ali, et al. Ms-iuffno: Multi-scale implicit u-net enhanced factorized fourier neural operator for solving geometric pdes. *Computer Methods in Applied Mechanics and Engineering*, 2025.

T. Lookman, P. V. Balachandran, D. Xue, and R. Yuan. Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design. *npj Computational Materials*, 5(1):21, 2019. doi: 10.1038/s41524-019-0153-8.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021a. doi: 10.1038/s42256-021-00302-5.

L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021b. doi: 10.1137/21M1397908.

L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022. doi: 10.1016/j.cma.2022.114778.

Luis Mandl, Somdatta Goswami, et al. Separable deeponet: Breaking the curse of dimensionality in physics-informed machine learning. *Computer Methods in Applied Mechanics and Engineering*, 2024. arXiv:2407.15887.

Z. Mao, L. Lu, O. Yeo, and G. E. Karniadakis. Deepm&mnet-pgd: A physics-guided deep learning framework for parametric conditional inference for compressible flows. *Computer Methods in Applied Mechanics and Engineering*, 414:116209, 2023. doi: 10.1016/j.cma.2023.116209.

N. Margenberg, C. Lessig, and T. Richter. Neural networks as preconditioners for iterative solvers in simulations of electrical machines. *IEEE Transactions on Magnetics*, 59(5):1–4, 2023. doi: 10.1109/TMAG.2023.3243521.

S. Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Frontiers in Big Data*, 4:669097, 2021. doi: 10.3389/fdata.2021.669097.

F. Masi, I. Stefanou, P. Vannucci, and V. Maffi-Berthier. Thermodynamics-based artificial neural networks for constitutive modeling. *Journal of the Mechanics and Physics of Solids*, 147:104277, 2021. doi: 10.1016/j.jmps.2020.104277.

X. Meng, L. Yang, Z. Mao, D. del Castillo-Negrete, and G. E. Karniadakis. Learning functional priors and posteriors from data and physics. *Journal of Computational Physics*, 457:111073, 2022. doi: 10.1016/j.jcp.2022.111073.

Taiki Miyagawa and Takeru Yokota. Physics-informed neural networks for functional differential equations: Cylindrical approximation and its convergence guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Viggo Moro and Luiz F. O. Chamon. Solving differential equations with constrained learning. In *International Conference on Learning Representations (ICLR)*, 2025.

B. Moseley, A. Markham, and T. Nissen-Meyer. Solving the wave equation with physics-informed deep learning. *arXiv preprint arXiv:2006.11894*, 2023a.

B. Moseley, T. Nissen-Meyer, and A. Markham. Deep learning for fast simulation of seismic waves in complex media. *Solid Earth*, 11:1527–1549, 2023b. doi: 10.5194/se-11-1527-2020.

J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, P. Hassanzadeh, K. Kashinath, and A. Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

B. Peherstorfer, D. Butnaru, K. Willcox, and H.-J. Bungartz. Localized discrete empirical interpolation method. *SIAM Journal on Scientific Computing*, 36(1):A168–A192, 2014. doi: 10.1137/130924408.

B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018. doi: 10.1137/16M1082469.

M. Penwarden, S. Zhe, A. Narayan, and R. M. Kirby. Multifidelity modeling for physics-informed neural networks (pinns). *Journal of Computational Physics*, 451:110844, 2023. doi: 10.1016/j.jcp.2021.110844.

T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. 2021.

A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. E. Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023. doi: 10.1016/j.jcp.2022.111902.

Shaoxiang Qin, Yuxuan Zhou, et al. Toward a better understanding of fourier neural operators: Analysis and improvement from a spectral perspective. *arXiv preprint arXiv:2404.07200*, 2024.

Rundi Qiu, Junzhe Li, Jingzhu Wang, Chun Fan, and Yiwei Wang. Direct numerical simulations of three-dimensional two-phase flow using physics-informed neural networks with a distributed parallel training algorithm. *Journal of Fluid Mechanics*, 2025.

A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*, volume 92 of *Unitext*. Springer, 2015. doi: 10.1007/978-3-319-15431-2.

N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. pp. 5301–5310, 2019.

M. Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19:1–24, 2018.

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*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020. doi: 10.1126/science.aaw4741.

C. Rao, H. Sun, and Y. Liu. Physics-informed deep learning for computational elastodynamics without labeled data. *Journal of Engineering Mechanics*, 147(8):04021043, 2021. doi: 10.1061/(ASCE)EM.1943-7889.0001947.

S. Rezaei, A. Harandi, A. Moeineddin, B.-X. Xu, and S. Reese. A mixed formulation for physics-informed neural networks as a potential solver for engineering problems in heterogeneous domains: Comparison with finite element method. *Computer Methods in Applied Mechanics and Engineering*, 401:115616, 2022. doi: 10.1016/j.cma.2022.115616.

G. Rozza, D. B. P. Huynh, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):229–275, 2008. doi: 10.1007/s11831-008-9019-9.

J. Ruf and W. Wang. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance*, 24(1):1–46, 2020.

F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020. doi: 10.3389/fphy.2020.00042.

E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362:112790, 2020. doi: 10.1016/j.cma.2019.112790.

A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. pp. 8459–8468, 2020.

V. Sekar, M. Zhang, C. Shu, and B. C. Khoo. Inverse design of airfoil using a deep convolutional neural network. *AIAA Journal*, 57(3):993–1003, 2019. doi: 10.2514/1.J057894.

J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. doi: 10.1016/j.jcp.2018.08.029.

I. Sosnovik and I. Oseledets. Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 34(4):215–223, 2019. doi: 10.1515/rnam-2019-0018.

B. Staber and S. Da Veiga. Conformal prediction for uncertainty quantification in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 418:116425, 2024. doi: 10.1016/j.cma.2023.116425.

M. Takamoto, T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, and M. Niepert. Pdebench: An extensive benchmark for scientific machine learning. 35:1596–1611, 2022.

K. Taneja, D. Kochkov, S. Augenstein, Z. Mao, and A. Anandkumar. Scalable neural network models for aerodynamic shape optimization. *arXiv preprint arXiv:2303.11128*, 2023.

M. Tang, Y. Liu, and L. J. Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics*, 413:109456, 2020. doi: 10.1016/j.jcp.2020.109456.

P. Thakolkaran, A. Joshi, Y. Zheng, M. Flaschel, L. De Lorenzis, and S. Kumar. Nn-euclid: Deep-learning hyperelasticity without stress data. *Journal of the Mechanics and Physics of Solids*, 169:105076, 2022. doi: 10.1016/j.jmps.2022.105076.

A. Tran, A. Mathews, L. Xie, and C. S. Ong. Factorized fourier neural operators. 2023.

R. K. Tripathy and I. Bilionis. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, 375:565–588, 2018. doi: 10.1016/j.jcp.2018.08.036.

J.-X. Wang, J. Wu, and H. Xiao. Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2:034603, 2017. doi: 10.1103/PhysRevFluids.2.034603.

S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40):eabi8605, 2021a. doi: 10.1126/sciadv.abi8605.

S. Wang, X. Yu, and P. Perdikaris. Competitive gradient descent. *arXiv preprint arXiv:2006.09059*, 2021b.

S. Wang, S. Sankaran, and P. Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022a.

S. Wang, X. Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022b. doi: 10.1016/j.jcp.2021.110768.

Sifan Wang, Ananyae Kumar Bhartari, Bowen Li, and Paris Perdikaris. Gradient alignment in physics-informed neural networks: A second-order optimization perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yizheng Wang, Jia Sun, Jinshuai Bai, Cosmin Anitescu, Mohammad Sadegh Eshaghi, Xiaoying Zhuang, Timon Rabczuk, and Yinghua Liu. Kolmogorov–arnold-informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on kolmogorov–arnold networks. *Computer Methods in Applied Mechanics and Engineering*, 2024.

P. R. Wiecha and O. L. Muskens. Deep learning meets nanophotonics: a generalized accurate predictor for near fields and far fields of arbitrary 3d nanostructures. *Nano Letters*, 20(1):329–338, 2020. doi: 10.1021/acs.nanolett.9b03971.

C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023a. doi: 10.1016/j.cma.2022.115671.

H. Wu, T. Hu, H. Luo, J. Wang, and M. Long. Solving high-dimensional pdes with latent spectral models. 2023b.

P. Wu, J. Sun, X. Chang, W. Zhang, R. Arcucci, Y. Guo, and C. C. Pain. Data-driven reduced order model with temporal convolutional neural network. *Computer Methods in Applied Mechanics and Engineering*, 360:112766, 2020. doi: 10.1016/j.cma.2019.112766.

L. Yang, X. Meng, and G. E. Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021. doi: 10.1016/j.jcp.2020.109913.

Y. Yang and P. Perdikaris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152, 2019. doi: 10.1016/j.jcp.2019.05.027.

Yahong Yang. Deeponet for solving nonlinear partial differential equations with physics-informed training. *arXiv preprint arXiv:2410.04344*, 2025. Revised September 2025.

A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS Computational Biology*, 16(11):e1007575, 2020. doi: 10.1371/journal.pcbi.1007575.

M. Yin, Z. Zheng, A. Anandkumar, and Y. Shu. Dimon: Learning solution operators of pdes on manifolds. *Nature Computational Science*, 2024. doi: 10.1038/s43588-024-00645-w. In press.

Y. Yin, M. Kirchmeyer, J.-Y. Franceschi, A. Rakotomamonjy, and P. Gallinari. Continuous pde dynamics forecasting with implicit neural representations. 2022.

Qingyang Zhang, Xiaowei Guo, Xinhai Chen, Chuanfu Xu, and Jie Liu. Par-deeponet: A novel physics informed operator learning method based on physical adaptive refinement. 2024.

S. Zhang and G. Lin. Robust data-driven discovery of governing physical laws with error bars. *Proceedings of the Royal Society A*, 474:20180305, 2018. doi: 10.1098/rspa.2018.0305.

T. Zhang, H. Xiao, and D. Ghosh. Physics-informed fourier neural operators for parametric pdes. *Journal of Nonlinear and Variational Analysis*, 9(1), 2025a.

Zhuo Zhang, Xiong Xiong, Sen Zhang, Wei Wang, Xi Yang, Shilin Zhang, and Canqun Yang. A pseudo-time stepping and parameterized physics-informed neural network framework for navier-stokes equations. *Physics of Fluids*, 37(3):033612, 2025b. doi: 10.1063/5.0259583.

Zhuo Zhang, Xiong Xiong, Sen Zhang, Wei Wang, Yanxu Zhong, Canqun Yang, and Xi Yang. Legendkinn: A legendre polynomial-based kolmogorov–arnold-informed neural network for efficient pde solving. *Expert Systems With Applications*, 2025c.

Jianwei Zheng, Liwei No, Ni Xu, Junwei Zhu, Xiaoxu Lin, and Xiaoqin Zhang. Alias-free mamba neural operator. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Weiheng Zhong and Hadi Meidani. Physics-informed discretization-independent deep compositional operator network. *Computer Methods in Applied Mechanics and Engineering*, 2024.

Weiheng Zhong and Hadi Meidani. Physics-informed geometry-aware neural operator. *Computer Methods in Applied Mechanics and Engineering*, 434:117540, 2025. doi: 10.1016/j.cma.2024.117540.

Juner Zhu, Yuyao Chu, Wenrui Liu, and Martin Z. Bazant. Phase-field deeponet: Physics-informed deep operator neural network for fast simulations of pattern formation governed by gradient flows of free-energy functionals. *Computer Methods in Applied Mechanics and Engineering*, 2023.

N. Zobeiry and K. D. Humfeld. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101:104232, 2021. doi: 10.1016/j.engappai.2021.104232.

Yiye Zou, Tianyu Li, Lin Lu, Jingyu Wang, Shufan Zou, Laiping Zhang, and Xiaogang Deng. Finite-difference-informed graph network for solving steady-state incompressible flows on block-structured grids. *Physics of Fluids*, 2025.