Neural Networks Meet Physics: A Survey of Physics-Informed Approaches to Modeling and Simulation

Karthika Nasir, Rahul Menon, Sneha Iyer

Neural Networks Meet Physics: A Survey of Physics-Informed Approaches to Modeling and Simulation

Karthika Nasir[1], Rahul Menon[2], and Sneha Iyer[3]

[1]Bharti Centre for Communication, Indian Institute of Technology Bombay, India
karthika.nasir@ee.iitb.ac.in
[2]Department of Computational Sciences, Indian Institute of Science, Bangalore, India
rahul.menon@iisc.ac.in
[3]Department of Electrical Engineering, Indian Institute of Technology Madras, India
sneha.iyer@ee.iitm.ac.in

Abstract. Physics-Informed Neural Networks (PINNs) have emerged as a powerful paradigm at the intersection of deep learning and computational physics, enabling the integration of prior physical knowledge into data-driven models. By embedding partial differential equations (PDEs), boundary conditions, and other physical constraints into the learning process, PINNs offer a flexible and mesh-free approach to solving forward and inverse problems in science and engineering. This survey provides a comprehensive and critical overview of the field, starting from foundational principles and mathematical formulations to recent theoretical advances and algorithmic innovations. We discuss various network architectures, training strategies, optimization challenges, and implementation practices that influence the performance and scalability of PINNs. A broad range of applications is reviewed, including fluid dynamics, solid mechanics, heat transfer, electromagnetics, biomedical modeling, and geoscientific problems. Particular attention is given to emerging directions such as probabilistic PINNs, operator learning, domain decomposition, and uncertainty quantification. We also examine the limitations of current approaches, including issues related to convergence, stiffness, and high-dimensional scalability. Finally, we outline future research avenues that could enhance the robustness, interpretability, and real-world applicability of PINNs. This review aims to serve as both an accessible introduction for newcomers and a detailed reference for experts seeking to advance the state of the art in physics-informed machine learning.

Keywords: Physics-Informed Neural Networks (PINNs); Scientific Machine Learning; Partial Differential Equations; Deep Learning; Inverse Problems; Computational Physics; Neural Operators; Surrogate Modeling; Physics-Based Constraints; Uncertainty Quantification.

1 Introduction

Over the past decade, the advent of deep learning has revolutionized a wide spectrum of scientific and engineering disciplines, yielding unprecedented progress in areas ranging from computer vision and natural language processing to healthcare diagnostics and autonomous systems. However, the application of neural networks to physical systems, governed by partial differential equations (PDEs) and other mathematical formulations, has traditionally been challenged by the need for large datasets, limited interpretability, and a lack of consistency with established physical laws. In response to these challenges, a novel paradigm known as *Physics-Informed Neural Networks* (PINNs) has emerged, offering a promising avenue to incorporate physical knowledge directly into the structure and training process of neural networks [1]. This paradigm shift enables neural networks to become not merely function approximators, but powerful tools for scientific discovery, modeling, and simulation. Physics-Informed Neural Networks represent a class of deep learning frameworks that leverage known physical laws—typically expressed in the form of differential equations—as soft or hard constraints during the training process. Unlike traditional data-driven approaches that rely solely on empirical observations, PINNs are guided by governing equations that describe the behavior of dynamical systems. This integration of data and physics allows for the construction of models that are more generalizable, data-efficient, and physically consistent. In essence, PINNs operate at the intersection of machine learning, computational physics, and applied mathematics, offering a unified framework to address both forward and inverse problems across a variety of disciplines. The foundational concept behind PINNs was introduced by Raissi, Perdikaris, and Karniadakis in a seminal series of papers beginning in 2017, where they demonstrated that neural networks could be trained not only to interpolate scattered data but also to satisfy the underlying physical laws in the form of PDE residuals. Since then, the field has witnessed rapid growth, with PINNs being extended and applied to a diverse array of problems, including fluid dynamics, solid mechanics, electromagnetics, geophysics, biophysics, and finance [2]. These applications range from solving high-dimensional PDEs and inferring hidden parameters to accelerating numerical solvers and discovering unknown governing laws from sparse observations [3]. The appeal of PINNs lies in several key advantages [4]. First, by embedding the physics into the loss function, PINNs reduce the dependence on large datasets, which are often expensive or impractical to obtain in scientific applications. Second, the learned models inherently respect the conservation laws and symmetries of the physical system, enhancing interpretability and robustness [5]. Third, PINNs offer a mesh-free approach to solving PDEs, thus avoiding the computational complexities associated with traditional numerical methods such as finite element or finite difference schemes. Furthermore, PINNs are highly flexible, capable of handling irregular domains, moving boundaries, multi-physics couplings, and parameter uncertainties within a unified learning framework [6]. Despite their potential, the practical deployment of PINNs is not without challenges [7]. The training of PINNs can be computationally intensive, particularly for stiff or multi-scale

PDEs. The choice of network architecture, activation functions, and loss weighting strategies significantly affects convergence and accuracy. Moreover, there is an ongoing need to understand the theoretical underpinnings of PINNs, such as approximation capabilities, error bounds, and stability properties [8]. Recent research has begun to address these issues through algorithmic innovations, such as adaptive sampling, domain decomposition, transfer learning, and incorporation of symmetries and conservation principles. Nevertheless, many open questions remain, and the field continues to evolve rapidly [9]. This survey aims to provide a comprehensive and critical overview of the rapidly expanding landscape of physics-informed neural networks [10]. We begin by reviewing the mathematical foundations of PINNs, including the formulation of forward and inverse problems, network architectures, and training methodologies. We then categorize and analyze the extensive range of applications across different domains of science and engineering, highlighting the strengths and limitations observed in practice [11]. Special attention is given to recent extensions of the PINN framework, such as stochastic PINNs, operator learning methods, and hybrid approaches combining physics-based and data-driven models. We also explore the growing body of theoretical work that seeks to formalize the capabilities and constraints of PINNs [12]. Furthermore, we discuss the computational aspects of implementing PINNs, including efficient solvers, automatic differentiation, hardware acceleration, and software frameworks that have been developed to facilitate their use. We identify key benchmarks and datasets that are commonly used for evaluation, and propose criteria for fair and reproducible comparison among different approaches [13]. Finally, we outline current research frontiers and open problems, offering insights into promising directions for future work. In summary, this review seeks to serve as both a tutorial introduction and an authoritative reference for researchers and practitioners interested in the theory, implementation, and application of physics-informed neural networks. By bridging the gap between deep learning and scientific modeling, PINNs hold the promise of transforming how we understand, simulate, and control complex physical systems in the era of data-driven science.

2 Mathematical Foundations and Core Principles of Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) are built upon a synergistic blend of classical mathematical modeling and modern machine learning. At the heart of this framework lies the use of deep neural networks (DNNs) as universal function approximators, constrained by governing physical laws expressed in the form of differential equations. In this section, we present the mathematical underpinnings that constitute the theoretical framework of PINNs, including the formulation of forward and inverse problems, the incorporation of PDE residuals into loss functions, and the use of automatic differentiation for gradient computation.

2.1 Differential Equations and Problem Settings

Many physical systems are governed by partial differential equations (PDEs), which describe the relationships between spatial and temporal variations of physical quantities. Consider a general PDE of the form:

$$\mathcal{N}[u](\mathbf{x}, t) = f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad t \in [0, T], \tag{1}$$

where $\mathcal{N}$ is a (possibly nonlinear) differential operator, $u(\mathbf{x}, t)$ is the solution field of interest, and $f(\mathbf{x}, t)$ is a known source term. The domain Ω is typically accompanied by initial and boundary conditions that fully specify the problem:

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{2}$$

$$u(\mathbf{x}, t) = g(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega [14]. \tag{3}$$

PINNs aim to approximate the solution $u(\mathbf{x}, t)$ by a neural network $u_\theta(\mathbf{x}, t)$ with parameters θ, trained to satisfy both the observational data and the physics encoded by the PDE.

2.2 Neural Network Approximation of the Solution

The core idea in PINNs is to model the unknown solution u using a deep neural network. The neural network u_θ takes as input the spatial and temporal coordinates $(\mathbf{x}, t)$ and outputs the corresponding prediction [15]. This approximation is trained using a composite loss function that enforces both data fidelity and PDE consistency:

$$\mathcal{L}(\theta) = \lambda_d \mathcal{L}_{\text{data}}(\theta) + \lambda_r \mathcal{L}_{\text{residual}}(\theta) + \lambda_b \mathcal{L}_{\text{boundary}}(\theta), \tag{4}$$

where $\lambda_d, \lambda_r, \lambda_b$ are weighting coefficients balancing the relative importance of different components [16].

Data Loss When observational data is available, it is incorporated via a mean squared error (MSE) loss on the difference between predicted and observed values:

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N_d} \sum_{i=1}^{N_d} \left| u_\theta(\mathbf{x}_i, t_i) - u_i^{\text{obs}} \right|^2, \tag{5}$$

where $\{(\mathbf{x}_i, t_i, u_i^{\text{obs}})\}_{i=1}^{N_d}$ denotes the set of observations [17].

Physics Loss via PDE Residuals To enforce physical consistency, the residual of the PDE is evaluated at a set of collocation points $\{(\mathbf{x}_j^r, t_j^r)\}_{j=1}^{N_r}$:

$$\mathcal{L}_{\text{residual}}(\theta) = \frac{1}{N_r} \sum_{j=1}^{N_r} \left| \mathcal{N}[u_\theta](\mathbf{x}_j^r, t_j^r) - f(\mathbf{x}_j^r, t_j^r) \right|^2. \tag{6}$$

These residuals are typically computed using automatic differentiation (AD), which enables exact derivatives with respect to the network inputs, avoiding the need for numerical differentiation.

Boundary and Initial Condition Loss Initial and boundary conditions are also enforced through suitable penalty terms:

$$\mathcal{L}_{\text{initial}}(\theta) = \frac{1}{N_0} \sum_{i=1}^{N_0} \left| u_\theta(\mathbf{x}_i^0, 0) - u_0(\mathbf{x}_i^0) \right|^2, \tag{7}$$

$$\mathcal{L}_{\text{boundary}}(\theta) = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| u_\theta(\mathbf{x}_i^b, t_i^b) - g(\mathbf{x}_i^b, t_i^b) \right|^2. \tag{8}$$

These losses ensure that the network respects prescribed physical constraints throughout training [18].

2.3 Automatic Differentiation and Computational Graphs

A critical enabler of PINNs is automatic differentiation (AD), which computes exact derivatives of neural network outputs with respect to inputs [19]. Unlike symbolic differentiation, which can be inefficient, or numerical differentiation, which is prone to instability, AD leverages the computational graph of the network to perform backpropagation of derivatives with machine precision [20]. This is particularly important when evaluating complex PDE residuals involving higher-order derivatives.

2.4 Forward vs. Inverse Problem Formulation

PINNs are inherently flexible and can be adapted to solve both forward and inverse problems. In the forward setting, the objective is to compute the solution u of a known PDE with given parameters and boundary conditions. In the inverse setting, certain parameters (e.g., coefficients, source terms, or initial states) are treated as unknowns and are learned by minimizing the discrepancy between observed data and the model's predictions. This dual capability makes PINNs a valuable tool for parameter identification, system calibration, and data assimilation.

2.5 Hard vs. Soft Constraint Imposition

PINNs may enforce physical constraints either softly, via penalization in the loss function, or hard, by explicitly embedding them into the network architecture. For example, known boundary conditions can be satisfied exactly by constructing trial functions that inherently fulfill those conditions (e.g., through function transformations or constrained architectures). The trade-off between soft and hard constraints remains an active area of research, particularly in problems where constraint satisfaction is critical.

2.6 Summary and Role of the Foundation in the PINN Pipeline

In summary, the mathematical formulation of PINNs revolves around the embedding of physical laws into the training of neural networks. By treating the governing equations as regularization terms, PINNs align the learning process with domain knowledge, thereby enhancing generalization and reliability [21]. The integration of differential operators, data loss, and constraint enforcement within a unified computational graph defines the core workflow of PINNs[22, 23]. These foundational principles form the basis upon which numerous extensions and applications have been developed, as we explore in the subsequent sections [24].

3 Applications of Physics-Informed Neural Networks

Since their introduction, Physics-Informed Neural Networks (PINNs) have demonstrated remarkable versatility across a wide range of scientific and engineering domains [25]. By embedding the governing equations of physical systems into neural networks, PINNs have enabled researchers to tackle problems that are otherwise difficult to solve with classical methods alone, especially in data-scarce or high-dimensional settings. This section surveys key application areas where PINNs have been employed, highlighting their unique advantages, methodological adaptations, and representative case studies.

3.1 Fluid Dynamics

One of the most prominent application areas for PINNs is computational fluid dynamics (CFD), where governing equations such as the Navier–Stokes equations describe the behavior of incompressible or compressible fluid flows:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \Delta \mathbf{u}, \tag{9}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{10}$$

Here, $\mathbf{u}$ is the velocity field, p is the pressure, and ν is the kinematic viscosity. PINNs have been used to:

- Solve forward simulations of flow over airfoils and bluff bodies.
- Infer hidden quantities such as pressure fields from velocity data.
- Reconstruct entire flow fields from sparse or partial measurements (e.g., in PIV experiments) [26].
- Model turbulent flow regimes using hybrid turbulence-PINN frameworks.

Notably, PINNs have shown promise in approximating high-fidelity solutions while being significantly less dependent on mesh resolution compared to traditional CFD solvers. However, their performance degrades in high Reynolds number regimes, motivating ongoing research on adaptive sampling, multi-network decompositions, and preconditioning strategies.

3.2 Solid Mechanics and Elasticity

In solid mechanics, PINNs have been applied to solve elasticity and elastoplasticity problems governed by the linear or nonlinear elasticity equations. Applications include:

- Stress-strain field reconstruction in heterogeneous materials [27].
- Fracture mechanics with crack tip singularities.
- Identification of material constitutive parameters in inverse settings.

For example, PINNs have been used to infer spatially varying Young's modulus in damaged tissues or composite structures from displacement data. These approaches have proven effective in biomechanics and structural health monitoring [28].

3.3 Heat Transfer and Thermofluid Systems

PINNs have also been deployed to solve transient and steady-state heat conduction problems, often modeled by the heat equation:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u, \tag{11}$$

where u is the temperature and α is the thermal diffusivity [29]. Applications include:

- Thermal tomography for reconstructing heat sources or initial conditions.
- Solving coupled thermo-fluid systems in microchannels and electronic cooling.
- Online temperature field prediction in heat exchangers or battery packs.

3.4 Electromagnetics and Acoustics

The solution of Maxwell's equations and Helmholtz-type problems has attracted considerable attention in electromagnetic wave propagation, antenna design, and acoustic modeling. PINNs have been explored to:

- Reconstruct electric and magnetic fields in waveguides or resonant cavities [30].
- Solve inverse scattering problems for radar and imaging systems.
- Model acoustic pressure fields in room acoustics or sonar applications.

3.5 Geophysics and Subsurface Modeling

Subsurface modeling involves inverse problems that are often ill-posed due to limited sensor coverage and noisy data. PINNs have been used to:

- Identify underground permeability and porosity fields.
- Solve forward problems in groundwater flow and seismic wave propagation.
- Perform inversion of geophysical data (e.g., seismic travel times or resistivity measurements) [31].

In many of these applications, PINNs serve as a data-physics fusion framework that balances model uncertainty with physical plausibility [32].

3.6 Biophysics and Physiology

In biological systems, PINNs offer an attractive approach to model processes that are governed by reaction-diffusion equations or nonlinear dynamical systems [33]. Examples include:

- Modeling calcium wave propagation in cardiac tissues.
- Estimating physiological parameters from medical imaging or electrophysiological recordings [34].
- Solving inverse problems in hemodynamics and vascular flow.

These applications leverage the capability of PINNs to integrate sparse measurements with complex, spatially variable PDE models [35].

3.7 Finance and Quantitative Modeling

In quantitative finance, PINNs have been used to solve stochastic differential equations (SDEs) and related PDEs such as the Black–Scholes or HJB equations. Applications include:

- Option pricing under uncertain volatility models [36].
- Portfolio optimization using dynamic programming formulations [37].
- Calibrating models to market data with embedded PDE constraints [38].

3.8 Other Emerging Areas

Beyond the domains discussed above, PINNs are being explored in:

- Climate modeling and atmospheric science.
- Quantum mechanics and Schrödinger-type equations [39].
- Control and robotics for modeling system dynamics and safety constraints.

These emerging applications point toward the growing generality and appeal of PINNs as tools for domain-aware machine learning.

3.9 Comparison with Traditional Solvers

In many cases, PINNs serve not only as alternatives but also as complementary tools to classical numerical solvers [40]. While traditional methods like finite difference, finite volume, and finite element methods provide rigorous discretizations with established error analysis, they often struggle with high-dimensional problems, complex geometries, and real-time inference tasks. PINNs, by contrast, offer:

- Mesh-free formulations [41].
- Native handling of irregular or moving boundaries.
- Built-in differentiability for sensitivity and adjoint analysis [42].
- Seamless integration of experimental data and physical models [43].

However, they remain sensitive to hyperparameters, suffer from training instabilities, and lack robust convergence guarantees, especially in stiff or multi-scale regimes [44]. Therefore, PINNs are often used in hybrid approaches where classical solvers provide coarse-grained solutions or priors, while PINNs refine or infer specific components [45].

3.10 Summary

The diversity of PINN applications reflects their broad utility in modeling, inference, and control of physical systems [6]. Across disciplines, the ability to incorporate domain knowledge into machine learning architectures provides a powerful inductive bias, enabling data-efficient, generalizable, and physically interpretable models [46]. The next section will explore recent extensions of the PINN framework that address current limitations and expand its applicability to more complex and multi-fidelity systems.

4 Extensions and Variants of Physics-Informed Neural Networks

While the original formulation of Physics-Informed Neural Networks (PINNs) provides a unified framework for integrating physical laws into machine learning models, several limitations have emerged, especially when dealing with high-dimensional, multi-scale, or stiff systems. In response, a variety of extensions and architectural innovations have been proposed to address specific challenges related to scalability, convergence, expressivity, and uncertainty quantification. This section surveys major developments in the PINN landscape, categorizing them into architectural enhancements, training strategies, hybrid models, and operator-based formulations.

4.1 Domain Decomposition and Parallelization: XPINNs and VPINNs

To address the scalability and training inefficiencies of standard PINNs in large or complex domains, researchers have introduced domain decomposition techniques that partition the computational domain into smaller subdomains. Each subdomain is then handled by an independent PINN, and continuity is enforced at the interfaces [47].

XPINNs Extended PINNs (XPINNs) partition the spatial and/or temporal domain and employ different neural networks in each subdomain [48]. This allows for:

- Parallel training of subdomain-specific networks.
- Local adaptation to sharp gradients or discontinuities.
- Scalability to higher-dimensional problems.

Continuity conditions are enforced through soft or hard constraints at the interfaces, and collocation points can be adaptively assigned per subdomain.

VPINNs Variational PINNs (VPINNs) modify the loss formulation using variational (weak) forms of the governing equations rather than strong forms [49]. This allows the integration of the residuals against test functions, which can improve stability and convergence:

$$\int_{\Omega} \mathcal{R}(u_\theta) v(\mathbf{x})\, d\mathbf{x} = 0, \quad \forall v \in \mathcal{V}, \tag{12}$$

where $\mathcal{R}$ denotes the residual, and $\mathcal{V}$ is a suitable test function space. VPINNs resemble Galerkin or Petrov-Galerkin methods and enable the incorporation of traditional finite element basis functions [50].

4.2 Stochastic and Probabilistic PINNs

To quantify model uncertainty and handle stochastic systems, several probabilistic extensions of PINNs have been proposed.

Bayesian PINNs Bayesian PINNs incorporate uncertainty in the model parameters by placing a prior distribution on the neural network weights or the PDE parameters and performing posterior inference using variational inference or Hamiltonian Monte Carlo. These models output predictive distributions rather than point estimates, providing credible intervals and uncertainty bounds.

PINNs for Stochastic PDEs In systems governed by stochastic PDEs (SPDEs), such as those with random coefficients or noise-driven sources, stochastic PINNs are designed to learn mappings from random input variables (e.g., ξ in generalized polynomial chaos) to solution statistics or realizations:

$$\mathcal{N}[u](\mathbf{x}, t; \xi) = f(\mathbf{x}, t; \xi). \tag{13}$$

Approaches include physics-informed generative models and moment-based PINNs, which compute expected values and variances directly from the governing laws.

4.3 Multi-Fidelity and Hybrid PINNs

In practical applications, data and models often exist at multiple levels of fidelity—e.g., coarse simulations, fine simulations, and experimental observations. Multi-fidelity PINNs seek to leverage these heterogeneous sources by:

- Constructing multi-branch networks with fidelity-specific submodels.
- Using transfer learning to initialize high-fidelity PINNs with pretrained low-fidelity models.
- Incorporating Gaussian Process surrogates as prior models combined with PINN correction terms [51].

Hybrid PINNs also integrate traditional solvers or numerical libraries into the training loop, using physics-based solvers for parts of the domain and PINNs for hard-to-model components.

4.4 Operator Learning and DeepONets

A significant development in the evolution of PINNs is the concept of learning operators rather than functions [52]. In this setting, the goal is to approximate a mapping $\mathcal{G} : f \mapsto u$, where u is the solution of a PDE given a source term or boundary condition f.

DeepONets Deep Operator Networks (DeepONets) approximate nonlinear operators using a dual-branch architecture: a branch net processes the input function (e.g., boundary data), and a trunk net processes the location. The output is a function value at a point, enabling:

- Generalization across different PDE conditions.
- Training on datasets of function mappings.
- Rapid inference in real-time applications [53].

When combined with physics-based loss terms, these become Physics-Informed DeepONets (PI-DeepONets), merging operator learning with physical constraints.

4.5 Adaptive Sampling and Curriculum Learning

PINNs often struggle to resolve sharp gradients, discontinuities, or multi-scale features. To mitigate these issues, several sampling strategies have been developed:

Adaptive Collocation Point Refinement Rather than sampling collocation points uniformly, adaptive methods place more points in regions with large residuals or gradient magnitudes [54]. Strategies include:

- Residual-based adaptive sampling (ReBAS).
- Importance sampling with residual weighting.
- Physics-informed active learning frameworks [55].

Curriculum Learning Curriculum learning organizes training from simpler to more complex tasks [56]. For instance, a PINN may first learn to satisfy boundary conditions before being trained on the full PDE residual. This staged training can significantly improve convergence and robustness [57].

4.6 Hard Constraint Imposition and Constrained Neural Architectures

In standard PINNs, initial and boundary conditions are typically enforced via soft constraints in the loss function. However, alternative strategies embed constraints directly into the network architecture, such as:

- Constructing trial functions that automatically satisfy boundary conditions (e.g., using function transformations).

– Designing constrained output layers or enforcing symmetry conditions [58].
– Using physics-informed basis functions within the network layers [59].

These approaches improve constraint satisfaction and reduce the burden on the optimizer.

4.7 Transfer Learning and Pretrained PINNs

To reduce training time and improve generalization, transfer learning methods have been adapted to PINNs. These include:

– Fine-tuning pretrained PINNs on similar but not identical PDE problems.
– Using pretrained PINNs as priors in Bayesian frameworks [60].
– Continual learning frameworks for PINNs in time-dependent or evolving systems.

4.8 Summary and Outlook

The ecosystem of Physics-Informed Neural Networks has rapidly diversified through a broad array of methodological innovations [61]. These extensions address key challenges in expressivity, computational efficiency, uncertainty quantification, and adaptability to complex geometries and dynamics. As the field matures, we expect to see increasing integration of these variants into unified, modular frameworks that can be tailored to specific problem classes. Moreover, many of these developments open new frontiers in combining classical numerical methods with machine learning, pushing the boundaries of data-driven scientific computing. In the following section, we examine the emerging theoretical understanding of PINNs, including error analysis, convergence guarantees, and approximation properties that form the mathematical foundation of this growing field.

5 Theoretical Foundations and Analysis of Physics-Informed Neural Networks

Despite the empirical success of Physics-Informed Neural Networks (PINNs) across a wide spectrum of scientific domains, their theoretical underpinnings remain an active area of research. Understanding why and when PINNs succeed—or fail—requires careful analysis from the perspectives of approximation theory, optimization landscapes, numerical analysis, and generalization error. This section provides an overview of recent developments in the theoretical analysis of PINNs, focusing on convergence guarantees, approximation properties, generalization bounds, and the effect of network architecture and PDE structure [27].

5.1 Approximation Capabilities of Neural Networks for PDE Solutions

The starting point of any theoretical justification of PINNs lies in the universal approximation theorem, which asserts that feedforward neural networks with sufficient width and nonlinearity can approximate continuous functions to arbitrary precision [50]. In the context of PINNs, these results have been extended to:

- **Solution Functions:** Given a PDE solution $u^*(\mathbf{x})$, neural networks can approximate $u^*(\mathbf{x})$ in the $\mathcal{L}^p$ norm, under mild smoothness assumptions.
- **Differential Operators:** Neural networks can also approximate the action of differential operators $\mathcal{N}[u]$ when derivatives are computed via automatic differentiation [62]. Recent work has analyzed how well neural networks preserve smoothness and regularity under differential constraints.

Furthermore, Sobolev training and extensions to Sobolev spaces have enabled rigorous analysis of convergence in norms that consider both function values and their derivatives, thereby aligning more closely with the residual minimization objective of PINNs.

5.2 Error Decomposition and Convergence Behavior

The convergence of PINNs can be analyzed by decomposing the total error into three primary components:

$$\|u_\theta - u^*\| \leq \underbrace{\|u^* - u_{\text{best}}\|}_{\text{Approximation Error}} + \underbrace{\|u_{\text{best}} - u_\theta\|}_{\text{Optimization Error}} + \underbrace{\text{Generalization Gap}}_{\text{Discretization / Sampling Error}} \quad [63]. \tag{14}$$

- **Approximation Error:** Determined by the expressive capacity of the neural network architecture.
- **Optimization Error:** Depends on the ability of gradient-based optimizers to find good minima of a highly non-convex loss landscape.
- **Generalization Gap:** Arises due to finite sampling of collocation points, impacting the discrepancy between empirical and true residuals [64].

Several theoretical studies have attempted to provide upper bounds on each of these terms [65]. In particular, the generalization error has been bounded using Rademacher complexity and covering number estimates, which relate to the network width, depth, and smoothness of the target function.

5.3 Impact of PDE Properties on PINN Performance

The difficulty of training PINNs is highly problem-dependent and sensitive to the structure of the underlying PDE. Several critical factors include:

- **Equation Type:** Elliptic problems (e.g., Poisson) are typically more stable than parabolic or hyperbolic ones [66].
- **Conditioning:** The conditioning of the differential operator plays a key role. Ill-conditioned operators (e.g., stiff systems or high-Pe problems) can lead to vanishing gradients and unstable training.
- **Boundary Layers and Singularities:** PINNs often fail to resolve steep gradients or singular solutions unless supported by specialized sampling or network architecture (e.g., localized basis functions) [67].

Recent studies have examined the spectral properties of the neural tangent kernel (NTK) associated with PINNs. These works suggest that PINNs tend to favor the learning of low-frequency modes first, a phenomenon known as the *spectral bias*. This can be advantageous in smooth problems but detrimental when the solution exhibits high-frequency components.

5.4 Well-Posedness and Consistency of the PINN Framework

A well-posed learning problem requires that a unique, stable solution exists and that the learning procedure is consistent with the governing physical law. Recent works have addressed:

- **Consistency:** Demonstrating that minimizing the PINN loss converges to the true PDE solution as the number of collocation points tends to infinity, assuming sufficient network expressivity.
- **Stability:** Analyzing how small perturbations in data or network weights affect the solution.
- **Uniqueness:** Showing that the PINN solution converges to the true solution under injectivity of the residual operator and satisfaction of boundary conditions [68].

These properties are highly dependent on the balance between data constraints and physics constraints in the PINN loss function. Underparameterization or insufficiently expressive networks can lead to inconsistent approximations, especially when the loss is dominated by soft boundary enforcement.

5.5 Effect of Loss Function Design and Normalization

The construction of the loss function in PINNs critically influences both the convergence and stability of training [69]. Several theoretical and empirical studies have highlighted issues such as:

- **Loss Imbalance:** The PDE residual, boundary, and initial condition losses may operate on vastly different scales, requiring careful normalization or dynamic weighting strategies (e.g., adaptive loss balancing, NTK-based reweighting).
- **Gradient Pathologies:** Stiff PDEs often exhibit loss landscapes with steep valleys and flat plateaus, making gradient-based optimization challenging.

– **Gradient Flow Dynamics:** Analysis of the dynamics of gradient descent on PINN loss landscapes reveals insights into mode learning, convergence rate, and early stopping behavior.

Methods such as the "gradient path method," residual-based adaptive sampling, and curriculum-based loss scheduling have been proposed to mitigate these issues, but theoretical guarantees remain limited to specific cases [70].

5.6 Generalization Theory and Sampling Analysis

One of the central questions in PINN theory is generalization: how well does the trained neural network satisfy the PDE in unseen regions of the domain?

– **Sampling Density:** Sparse or poorly distributed collocation points can lead to underfitting in certain subregions, particularly in high-dimensional domains [71].
– **Coverage and Discrepancy:** Using concepts from quasi-Monte Carlo theory and discrepancy measures, researchers have established that the error in residual minimization can be bounded in terms of collocation coverage.
– **Learning Theory:** PAC-style bounds have been derived using tools from statistical learning theory, though they are often pessimistic or loose due to the high capacity of neural networks.

Adaptive and importance sampling strategies are partially motivated by these results, as they aim to reduce variance and increase effective sample complexity in critical regions.

5.7 Open Problems and Theoretical Challenges

Despite recent progress, the theoretical analysis of PINNs remains incomplete, with several major open questions:

– What are tight and practically meaningful convergence rates for PINNs in various PDE classes?
– How can one characterize the stability and conditioning of PINNs in the presence of noise or parameter uncertainty?
– Can generalization theory be made less pessimistic by incorporating structure from the PDE or geometry of the solution manifold?
– How do architectural choices (depth, activation, implicit layers) impact approximation error and convergence behavior in a provable manner?

Addressing these questions will not only improve our theoretical understanding but also guide the design of next-generation PINN architectures and training procedures [72].

5.8 Summary

In summary, the theoretical landscape of PINNs is rapidly evolving. While foundational results on approximation and convergence have provided critical insights, many practical aspects remain poorly understood from a theoretical perspective. Bridging the gap between empirical success and formal guarantees will require a cross-disciplinary effort involving approximation theory[73, 74], PDE analysis, numerical methods, and statistical learning theory [75]. In the next section, we shift our focus to implementation aspects and computational considerations that are crucial for deploying PINNs in real-world scientific computing applications.

6 Computational and Implementation Aspects

Although Physics-Informed Neural Networks (PINNs) are conceptually elegant, their practical implementation presents numerous challenges. The success of PINNs in real-world applications depends not only on the mathematical formulation, but also on various computational aspects including network architecture design, optimization techniques, hardware utilization, and integration with existing numerical software [76]. This section explores key practical considerations, offering a comprehensive view of the implementation strategies, software frameworks, and performance bottlenecks in deploying PINNs effectively.

6.1 Network Architecture Design

Feedforward Architectures Most classical PINNs adopt fully connected feedforward architectures (also known as multilayer perceptrons, or MLPs), typically with 3–10 hidden layers and 20–100 neurons per layer. Common activation functions include tanh, sigmoid, ReLU, or sine-based activations. The tanh activation remains popular due to its smoothness and differentiability, which are advantageous for gradient computations of differential operators [77].

Specialized Architectures Several architecture variants have been proposed to improve expressiveness or inductive bias:

- **Fourier Feature Networks:** Introduce sinusoidal inputs with high-frequency components to better represent fine-scale structures.
- **Residual Networks (ResNets):** Facilitate gradient flow and stabilize training in deeper models.
- **Adaptive Activation Networks:** Incorporate trainable parameters in the activation functions to dynamically adjust activation sharpness.
- **Constrained Networks:** Architectures explicitly designed to satisfy boundary or geometric constraints (e.g., via output transformations).

6.2 Optimization and Training Strategies

Loss Function Composition The typical PINN loss function includes terms for:

1. The PDE residual at collocation points.
2. Boundary and initial condition enforcement.
3. (Optionally) Supervised data fitting.

Training involves minimizing a weighted combination of these terms:

$$\mathcal{L}_{\text{total}} = \lambda_r \mathcal{L}_{\text{residual}} + \lambda_b \mathcal{L}_{\text{BC/IC}} + \lambda_d \mathcal{L}_{\text{data}}.$$

Gradient-Based Optimization PINNs are typically trained using gradient-based optimizers:

- **First-order methods:** Adam is commonly used due to its robustness to noisy gradients.
- **Second-order methods:** L-BFGS is often applied after initial training to accelerate convergence to local minima.

A popular hybrid approach uses Adam for pretraining and L-BFGS for fine-tuning [78].

Training Pathologies Common issues encountered during training include:

- **Vanishing gradients:** Especially in deep networks or stiff PDEs [79].
- **Unbalanced loss magnitudes:** Residual terms often dominate due to scale mismatch.
- **Poor convergence:** Sensitive to initialization, learning rates, and sampling strategies [80].

Solutions include dynamic loss weighting (e.g., NTK-based scaling), gradient normalization, adaptive sampling, and staged training schedules [81].

6.3 Sampling and Discretization

Collocation Point Generation The choice of collocation points $\{\mathbf{x}_i\}_{i=1}^{N_r}$ significantly affects generalization and accuracy:

- **Uniform grids:** Simple but may underrepresent complex regions.
- **Random sampling:** Reduces aliasing and bias but introduces variance [82].
- **Adaptive methods:** Focus sampling in regions of high residual or error.

Quadrature Techniques For weak-form PINNs or variational formulations, numerical integration is required to evaluate residuals. Methods include:

- Gaussian quadrature in 1D/2D domains [83].
- Monte Carlo and quasi-Monte Carlo integration in high dimensions [84].

6.4 Software Frameworks and Libraries

Several open-source software frameworks have been developed to facilitate PINN research and deployment:

- **DeepXDE:** A high-level Python library built on TensorFlow, offering support for forward/inverse problems and custom PDEs [85].
- **PINN-PyTorch:** Community-developed modules compatible with PyTorch, emphasizing flexibility and integration.
- **Modulus (NVIDIA):** A high-performance, industrial-grade library supporting multi-GPU training, complex geometries, and surrogate modeling.
- **NeuroDiffEq:** A lightweight PyTorch-based tool for educational and prototyping use.

These libraries support features such as automatic differentiation, boundary encoding, domain meshing, and distributed training.

6.5 Hardware and Computational Efficiency

Resource Requirements While PINNs do not require mesh generation like finite element methods, they can be computationally intensive due to:

- High number of collocation points.
- Automatic differentiation overhead for higher-order derivatives.
- Large network parameter space and slow convergence [2].

Acceleration Strategies Efficiency can be improved via:

- **GPU acceleration:** Essential for training deep or wide networks.
- **Parallel domain decomposition:** Used in XPINNs or multiphysics systems.
- **Mixed precision training:** Reduces memory footprint and speeds up computation.
- **Compiler optimization:** Using JAX or TorchScript for optimized derivative computation [86].

6.6 Benchmarking Practices

Standard Test Problems Common benchmark PDEs include:

- 1D Burgers' equation (nonlinear hyperbolic).
- 2D heat/diffusion equations (parabolic).
- 2D Poisson or Laplace equations (elliptic).
- Allen–Cahn and Navier–Stokes equations (nonlinear/stiff) [87].

These provide controlled settings to evaluate convergence, expressiveness, and robustness across architectures and sampling strategies.

Metrics and Diagnostics Key metrics include:

- Relative $\mathcal{L}_2$ error.
- PDE residual norms.
- Constraint satisfaction scores (for BC/IC) [88].
- Training time and memory usage.

Diagnostic tools such as loss surface visualization, eigenvalue analysis of the NTK, and gradient distribution plots are increasingly used to monitor and interpret PINN training dynamics [89].

6.7 Summary

The implementation of PINNs involves a delicate interplay between architectural choices, optimization dynamics, computational efficiency, and software infrastructure. Effective deployment requires careful tuning of hyperparameters, loss functions, and sampling strategies, supported by specialized libraries and increasingly powerful hardware. Despite these challenges, ongoing innovations in network design, domain decomposition, and parallelization continue to enhance the scalability and applicability of PINNs to increasingly complex physical systems. The next section provides a comprehensive overview of PINN applications across diverse scientific and engineering domains, demonstrating their potential as a universal modeling paradigm for forward and inverse problems governed by physical laws [90].

7 Conclusion and Future Perspectives

Physics-Informed Neural Networks (PINNs) represent a significant paradigm shift in scientific machine learning, enabling the unification of data-driven learning with domain-specific physical laws. By embedding governing equations directly into the training objective of neural networks, PINNs have shown remarkable potential in addressing forward and inverse problems across a wide spectrum of disciplines—from fluid dynamics and solid mechanics to biomedical modeling and geophysics. This review has provided a comprehensive survey of the theoretical foundations, architectural innovations, computational strategies, and real-world applications of PINNs. We have highlighted how PINNs leverage automatic differentiation, collocation sampling, and variational formulations to provide mesh-free, data-efficient alternatives to traditional numerical methods [91]. Despite these promising developments, PINNs remain an active and rapidly evolving field, facing several significant challenges. Chief among them is the issue of scalability: training PINNs on high-dimensional, stiff, or multiphysics problems is computationally demanding and often suffers from slow convergence or poor accuracy [92]. The issue of generalization also persists, particularly in regions of the domain that are under-sampled or where the solution exhibits sharp gradients or discontinuities. Moreover, while PINNs offer a natural solution for

problems with limited data, their reliance on a well-posed PDE model can become a limitation in highly uncertain or chaotic systems where the governing physics is only partially known or poorly specified [93]. The future development of PINNs will likely involve synergistic advances across multiple fronts:

- **Theory and expressivity:** A deeper theoretical understanding of approximation error, generalization bounds, and convergence behavior is essential [94]. Recent work on Neural Tangent Kernels (NTKs) and Sobolev training offers new insights but remains limited to simplified settings.
- **Architectural innovation:** Designing physics-informed architectures with better inductive bias—such as convolutional PINNs, graph-based PINNs, Fourier neural operators, and adaptive basis networks—will be key to improving efficiency and scalability.
- **Optimization and training:** Overcoming gradient pathologies, stiffness, and ill-conditioning in the optimization landscape is an urgent priority. Techniques such as curriculum learning, residual-based adaptive sampling, meta-learning, and second-order optimizers show promise but require further refinement [95].
- **Uncertainty quantification:** The integration of probabilistic modeling and Bayesian inference within the PINN framework is an emerging direction. Probabilistic PINNs (e.g., Bayesian PINNs, ensemble PINNs, variational PINNs) aim to quantify epistemic and aleatoric uncertainty, crucial for risk-sensitive applications [96].
- **Multiphysics and multiscale systems:** Extending PINNs to handle coupled systems and hierarchical models remains an open challenge. Hybrid models that integrate PINNs with finite element solvers, surrogate models, or reduced-order models offer a path forward.
- **Software and deployment:** The development of robust, user-friendly software libraries—such as DeepXDE, Modulus, and PINN-based extensions in TensorFlow or PyTorch—will accelerate adoption. Efficient deployment on GPUs, TPUs, and HPC environments will also be critical for real-world impact [97].

As the field matures, interdisciplinary collaboration will play an increasingly vital role. Engineers, physicists, and computer scientists must work together to refine problem formulations, identify application-specific constraints, and validate results against experimental or high-fidelity numerical benchmarks [98]. Moreover, the fusion of data-driven and physics-based modeling paradigms should not be seen as a dichotomy but rather as a continuum of approaches, tailored to the fidelity of available data and the confidence in governing equations [99].

In summary, PINNs stand at the intersection of deep learning and computational physics, offering a flexible and powerful framework for solving complex physical systems with embedded inductive bias. While numerous challenges remain, the rapid progress in both theoretical understanding and practical deployment suggests a promising future. As computational resources become more

powerful and domain knowledge is more seamlessly integrated into learning architectures, PINNs may become indispensable tools in the scientist's and engineer's toolkit, revolutionizing how we model, simulate, and understand the physical world.

References

1. Shangshang Yang, Ye Tian, Cheng He, Xingyi Zhang, Kay Chen Tan, and Yaochu Jin. A gradient-guided evolutionary approach to training deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4861–4875, 2021.
2. Yang Liu, Wen Liu, Xunshi Yan, Shuaiqi Guo, and Chen-an Zhang. Adaptive transfer learning for pinn. *Journal of Computational Physics*, 490:112291, 2023.
3. Yun Zou, Yifeng Zeng, Shuying Li, and Quing Zhu. Machine learning model with physical constraints for diffuse optical tomography. *Biomedical optics express*, 12(9):5720–5735, 2021.
4. Maryam Toloubidokhti, Yubo Ye, Ryan Missel, Xiajun Jiang, Nilesh Kumar, Ruby Shrestha, and Linwei Wang. Dats: Difficulty-aware task sampler for meta-learning physics-informed neural networks. In *The Twelfth International Conference on Learning Representations*, 2023.
5. Maziar Raissi, Paris Perdikaris, and George Em 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.
6. Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.
7. Rafael Bischof and Michael Kraus. Multi-objective loss balancing for physics-informed deep learning. *arXiv preprint arXiv:2110.09813*, 2021.
8. Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.
9. Mohammadamin Mahmoudabadbozchelou and Safa Jamali. Rheology-informed neural networks (rhinns) for forward and inverse metamodelling of complex fluids. *Scientific reports*, 11(1):1–13, 2021.
10. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.
11. Lulu Cao, Yufei Liu, Zhenzhong Wang, Dejun Xu, Kai Ye, Kay Chen Tan, and Min Jiang. An interpretable approach to the solutions of high-dimensional partial differential equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20640–20648, 2024.
12. Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024.
13. Qineng Wang, Liming Song, Zhendong Guo, Jun Li, and Zhenping Feng. A novel multi-fidelity surrogate for efficient turbine design optimization. *Journal of Turbomachinery*, 146(4), 2024.

14. Xiaonan Lai, Shuo Wang, Zhenggang Guo, Chao Zhang, Wei Sun, and Xueguan Song. Designing a shape–performance integrated digital twin based on multiple models and dynamic data: a boom crane example. *Journal of Mechanical Design*, 143(7):071703, 2021.
15. Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.
16. Chayan Banerjee, Kien Nguyen, Clinton Fookes, and Karniadakis George. Physics-informed computer vision: A review and perspectives. *ACM Computing Surveys*, 57(1):1–38, 2024.
17. Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Zhiwei Fang, Max Rietmann, Wonmin Byeon, and Sanjay Choudhry. Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In *Computational Science–ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part V*, pages 447–461. Springer, 2021.
18. Haixu Wu, Huakun Luo, Yuezhou Ma, Jianmin Wang, and Mingsheng Long. Ropinn: Region optimized physics-informed neural networks. *arXiv preprint arXiv:2405.14369*, 2024.
19. Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
20. Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8):11618–11633, 2020.
21. Chen Xu, Ba Trung Cao, Yong Yuan, and Günther Meschke. Transfer learning based physics-informed neural networks for solving inverse problems in tunneling. *arXiv e-prints*, pages arXiv–2205, 2022.
22. Binghang Lu, Christian Moya, and Guang Lin. Nsga-pinn: a multi-objective optimization method for physics-informed neural network training. *Algorithms*, 16(4):194, 2023.
23. Yassine Zniyed, Thanh Phuong Nguyen, et al. Efficient tensor decomposition-based filter pruning. *Neural Networks*, 178:106393, 2024.
24. Mohannad Elhamod, Jie Bu, Christopher Singh, Matthew Redell, Abantika Ghosh, Viktor Podolskiy, Wei-Cheng Lee, and Anuj Karpatne. Cophy-pgnn: Learning physics-guided neural networks with competing loss functions for solving eigenvalue problems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2020.
25. Wei Peng, Weien Zhou, Jun Zhang, and Wen Yao. Accelerating physics-informed neural network training with prior dictionaries. 2020.
26. Handing Wang, Yaochu Jin, and John Doherty. Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems. *IEEE transactions on cybernetics*, 47(9):2664–2677, 2017.
27. Abdul Hannan Mustajab, Hao Lyu, Zarghaam Rizvi, and Frank Wuttke. Physics-informed neural networks for high-frequency and multi-scale problems using transfer learning. *Applied Sciences*, 14(8):3204, 2024.
28. Kejun Tang, Xiaoliang Wan, and Chao Yang. Das-pinns: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, 2023.
29. Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica (Slovenia)*, 44(3), 2020.
30. Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

31. Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, November 2019.
32. Joaquim Peiró and Spencer Sherwin. Finite difference, finite element and finite volume methods for partial differential equations. *Handbook of Materials Modeling: Methods*, pages 2415–2446, 2005.
33. Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.
34. Yi Xiong, Pham Luu Trung Duong, Dong Wang, Sang-In Park, Qi Ge, Nagarajan Raghavan, and David W Rosen. Data-driven design space exploration and exploitation for design for additive manufacturing. *Journal of Mechanical Design*, 141(10):101101, 2019.
35. Taniya Kapoor, Hongrui Wang, Alfredo Núñez, and Rolf Dollevoet. Transfer learning for improved generalizability in causal physics-informed neural networks for beam simulations. *Engineering Applications of Artificial Intelligence*, 133:108085, 2024.
36. Jian Cheng Wong, Pao-Hsiung Chiu, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. Lsa-pinn: Linear boundary connectivity loss for solving pdes on complex geometry. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2023.
37. Florian Arnold and Rudibert King. State–space modeling for control based on physics-informed neural networks. *Engineering Applications of Artificial Intelligence*, 101:104195, 2021.
38. Apostolos F Psaros, Kenji Kawaguchi, and George Em Karniadakis. Meta-learning pinn loss functions. *Journal of computational physics*, 458:111121, 2022.
39. Zhao Wei, Jian Cheng Wong, Nicholas Wei Yong Sung, Abhishek Gupta, Chin Chun Ooi, Pao-Hsiung Chiu, My Ha Dao, and Yew-Soon Ong. How to select physics-informed neural networks in the absence of ground truth: a pareto front-based strategy. In *1st Workshop on the Synergy of Scientific and Machine Learning Modeling@ ICML2023*, 2023.
40. Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.
41. Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.
42. Stefano Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Frontiers in big Data*, page 92, 2021.
43. Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.
44. Xiaodong Cui, Wei Zhang, Zoltán Tüske, and Michael Picheny. Evolutionary stochastic gradient descent for optimization of deep neural networks. *Advances in neural information processing systems*, 31, 2018.
45. Risto Miikkulainen and Stephanie Forrest. A biological perspective on evolutionary computation. *Nature Machine Intelligence*, 3(1):9–15, 2021.
46. Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation*, 23(3):442–458, 2018.
47. Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized physics-informed neural networks for parameterized pdes. *arXiv preprint arXiv:2408.09446*, 2024.

48. Olga Fuks and Hamdi 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), 2020.
49. Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *Ieee Access*, 8:42200–42216, 2020.
50. Michael Penwarden, Shandian Zhe, Akil Narayan, and Robert M Kirby. A metalearning approach for physics-informed neural networks (pinns): Application to parameterized pdes. *Journal of Computational Physics*, 477:111912, 2023.
51. George Barbastathis, Aydogan Ozcan, and Guohai Situ. On the use of deep learning for computational imaging. *Optica*, 6(8):921–943, 2019.
52. Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):6136, 2021.
53. Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019.
54. Franz M Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C Geiger. On the apparent pareto front of physics-informed neural networks. *IEEE Access*, 2023.
55. Guangyong Sun, Fengxiang Xu, Guangyao Li, and Qing Li. Crashing analysis and multiobjective optimization for thin-walled structures with functionally graded thickness. *International Journal of Impact Engineering*, 64:62–74, 2014.
56. Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Journal of Machine Learning Research*, 18(18):1–65, 2017.
57. Abhishek Gupta and Yew-Soon Ong. *Memetic computation: the mainspring of knowledge transfer in a data-driven optimization era*, volume 21. Springer, 2018.
58. Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.
59. Shengze Cai, He Li, Fuyin Zheng, Fang Kong, Ming Dao, George Em Karniadakis, and Subra Suresh. Artificial intelligence velocimetry and microaneurysm-on-a-chip for three-dimensional analysis of blood flow in physiology and disease. *Proceedings of the National Academy of Sciences*, 118(13):e2100697118, 2021.
60. Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE transactions on neural networks and learning systems*, 34(2):550–570, 2021.
61. Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, and Yew-Soon Ong. Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, 5(3):985–1000, 2022.
62. Zhenkun Wang, Qingfu Zhang, Yew-Soon Ong, Shunyu Yao, Haitao Liu, and Jianping Luo. Choose appropriate subproblems for collaborative modeling in expensive multiobjective optimization. *IEEE Transactions on Cybernetics*, 53(1):483–496, 2021.
63. Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *Artificial Neural Networks—ICANN 2001: International Conference Vienna, Austria, August 21–25, 2001 Proceedings 11*, pages 87–94. Springer, 2001.
64. Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

65. Roger Temam. *Navier–Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Society, 2024.
66. Abhishek Gupta, Lei Zhou, Yew-Soon Ong, Zefeng Chen, and Yaqing Hou. Half a dozen real-world applications of evolutionary multitasking, and more. *IEEE Computational Intelligence Magazine*, 17(2):49–66, 2022.
67. Nils Wandel, Michael Weinmann, Michael Neidlin, and Reinhard Klein. Spline-pinn: Approaching pdes without data using fast, physics-informed hermite-spline cnns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8529–8538, 2022.
68. Zixue Xiang, Wei Peng, Xu Liu, and Wen Yao. Self-adaptive loss balanced physics-informed neural networks. *Neurocomputing*, 496:11–34, 2022.
69. John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.
70. Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, Pao-Hsiung Chiu, Joshua Shao Zheng Low, My Ha Dao, and Yew-Soon Ong. The baldwin effect in advancing generalizability of physics-informed neural networks. *arXiv preprint arXiv:2312.03243*, 2024.
71. Yu Xue, Yiling Tong, and Ferrante Neri. An ensemble of differential evolution and adam for training feed-forward neural networks. *Information Sciences*, 608:453–471, 2022.
72. Gargya Gokhale, Bert Claessens, and Chris Develder. Physics informed neural networks for control oriented thermal modeling of buildings. *Applied Energy*, 314:118852, 2022.
73. Yassine Zniyed, Thanh Phuong Nguyen, et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.
74. Suchuan Dong and Jielin Yang. On computing the hyperparameter of extreme learning machines: Algorithm and application to computational pdes, and comparison with classical and high-order finite elements. *Journal of Computational Physics*, 463:111290, 2022.
75. Yihang Gao, Ka Chun Cheung, and Michael K Ng. Svd-pinns: Transfer learning of physics-informed neural networks via singular value decomposition. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1443–1450. IEEE, 2022.
76. Zongyi Li. Neural operator: Learning maps between function spaces. In *2021 Fall Western Sectional Meeting*. AMS, 2021.
77. Bo Zhang and Chao Yang. Discovering physics-informed neural networks model for solving partial differential equations through evolutionary computation. *Swarm and Evolutionary Computation*, 88:101589, 2024.
78. Sebastian Flennerhag, Pablo G Moreno, Neil D Lawrence, and Andreas Damianou. Transferring knowledge across learning processes. *arXiv preprint arXiv:1812.01054*, 2018.
79. Caio Davi and Ulisses Braga-Neto. Pso-pinn: physics-informed neural networks trained with particle swarm optimization. *arXiv preprint arXiv:2202.01943*, 2022.
80. James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

81. Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.
82. Caio Davi and Ulisses Braga-Neto. Multi-objective pso-pinn. In *1st Workshop on the Synergy of Scientific and Machine Learning Modeling@ ICML2023*, 2023.
83. Shengze Cai, Zhicheng Wang, Frederik Fuest, Young Jin Jeon, Callum Gray, and George Em Karniadakis. Flow over an espresso cup: inferring 3-d velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks. *Journal of Fluid Mechanics*, 915:A102, 2021.
84. Rongye Shi, Zhaobin Mo, and Xuan Di. Physics-informed deep learning for traffic state estimation: A hybrid paradigm informed by second-order traffic models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 540–547, 2021.
85. Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.
86. Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021.
87. Sina Amini Niaki, Ehsan Haghighat, Trevor Campbell, Anoush Poursartip, and Reza Vaziri. Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. *Computer Methods in Applied Mechanics and Engineering*, 384:113959, 2021.
88. Elliott Skomski, Ján Drgoňa, and Aaron Tuor. Automating discovery of physics-informed neural state space models via learning and evolution. In *Learning for Dynamics and Control*, pages 980–991. PMLR, 2021.
89. Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.
90. Taco de Wolff, Hugo Carrillo Lincopi, Luis Martí, and Nayat Sanchez-Pi. Mopinns: an evolutionary multi-objective approach to physics-informed neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 228–231, 2022.
91. Xiwang He, Liangliang Yang, Zhuangzhuang Gong, Yong Pang, Jianji Li, Ziyun Kan, and Xueguan Song. Digital twin-based online structural optimization? yes, it's possible! *Thin-Walled Structures*, page 112796, 2024.
92. François Mazé and Faez Ahmed. Diffusion models beat gans on topology optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 9108–9116, 2023.
93. Woojin Cho, Kookjin Lee, Donsub Rim, and Noseong Park. Hypernetwork-based meta-learning for low-rank physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.
94. Raphaël Pellegrin, Blake Bullwinkel, Marios Mattheakis, and Pavlos Protopapas. Transfer learning with physics-informed neural networks for efficient simulation of branched flows. *arXiv preprint arXiv:2211.00214*, 2022.
95. Alex Bihlo. Improving physics-informed neural networks with meta-learned optimization. *Journal of Machine Learning Research*, 25(14):1–26, 2024.

96. Tian Qin, Alex Beatson, Deniz Oktay, Nick McGreivy, and Ryan P Adams. Meta-pde: Learning to solve pdes quickly without a mesh. *arXiv preprint arXiv:2211.01604*, 2022.
97. Lulu Cao, Zexin Lin, Kay Chen Tan, and Min Jiang. Interpretable solutions for multi-physics pdes using t-nngp. 2025.
98. Ehsan Haghighat, Sahar Abouali, and Reza Vaziri. Constitutive model characterization and discovery using physics-informed deep learning. *Engineering Applications of Artificial Intelligence*, 120:105828, 2023.
99. Tomoharu Iwata, Yusuke Tanaka, and Naonori Ueda. Meta-learning of physics-informed neural networks for efficiently solving newly given pdes. *arXiv preprint arXiv:2310.13270*, 2023.