A Review on Surrogate Modelling and Uncertainty Quantification of Neural Differential Equations

Vignesh Gopakumar

PhD Upgrade Viva Report

Centre for Artificial Intelligence, Department of Computer Science

University College London

July 25, 2025

Contents

Chapter 1

Introduction

Partial differential equations (PDEs) form the mathematical foundation for describing complex physical systems across numerous scientific and engineering domains. From fluid dynamics [1] and heat transfer [2] to nuclear fusion [3] and climate modelling [4], these equations capture the underlying physics that govern natural phenomena. Scientific computing and simulation have become indispensable tools for solving these equations, enabling researchers to explore complex systems under various conditions without the constraints of physical experimentation. However, computational solutions to PDEs, particularly for complex geometries and multiphysics applications, present significant challenges and a consequential carbon footprint [5]. These challenges are especially prominent in grand research areas like fusion energy, where multiple physics phenomena interact across different scales in intricate ways.

1.0.1 Traditional Numerical Methods and their Limitations

Traditional numerical methods such as finite difference [6], finite volume[7], finite element[8], and spectral methods [9], while mathematically rigorous, often require supercomputing resources and can take hours, days, or even weeks to deliver solutions [10], severely limiting their utility for iterative design exploration and real-time applications.

Consider modelling magnetohydrodynamics (MHD) in fusion reactors: classical methods for multi-physics and multi-scale simulation are only accurate when flow features are smooth, requiring meshes to resolve the smallest features. For complex systems like fusion reactors or climate models, this makes direct numerical simulation computationally prohibitive. The traditional approach involves using simplified versions of the governing equations to allow coarser meshes, sacrificing accuracy for feasibility. This resolution-efficiency tradeoff presents a significant bottleneck to scientific progress.

1.0.2 The Need for Surrogates

The quest to accelerate the solution of these equations has become increasingly urgent, particularly in domains like nuclear fusion, where rapid advancements are essential to meet ambitious energy production timelines. With insufficient time or funding to deliver commercial fusion through conventional 'test-based' development, our simulation capability must be transformed into an 'actionable' or predictive capability built upon state-of-the-art data+model hybrid simulation. This transformation necessitates a paradigm shift from traditional numerical solvers to more efficient computational approaches.

Neural surrogate models have emerged as a promising alternative, offering the potential to reduce computational costs by several orders of magnitude [11, 12]. These models, trained on data generated from high-fidelity simulations or experimental observations, learn to emulate the PDE solutions across various initial conditions and parameter configurations. Recent advancements in neural operators, graph neural networks, and other deep learning architectures have demonstrated remarkable capabilities in capturing complex PDE dynamics while maintaining computational efficiency. For instance, neural PDE solvers in fusion applications have shown speedups of six orders of magnitude over traditional numerical solvers while maintaining acceptable accuracy levels [13].

1.0.3 Actionable Surrogate Models

However, the transition from traditional numerical methods to neural surrogate models introduces its own set of challenges, particularly regarding the reliability and trustworthiness of these models. Unlike their numerical counterparts, which are built on well-established mathematical principles with known error bounds,

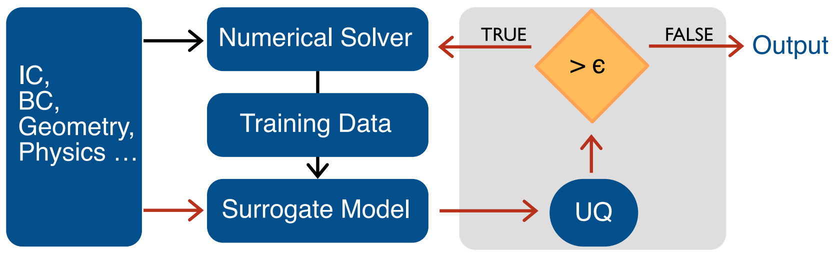

Figure 1.1: Neural PDE solvers use data from traditional numerical solvers to quickly approximate PDEs across various conditions (shown by black arrows). To ensure reliability, these models incorporate uncertainty quantification (UQ). If the predicted error exceeds a threshold ε, the numerical solver is used and added to the training data; otherwise, predictions are used as output (shown by red arrows).

neural surrogates may produce confident predictions without adequate uncertainty quantification (UQ), potentially leading to catastrophic consequences in safety-critical applications.

An actionable surrogate is one that can be confidently deployed in scientific and engineering workflows, with well-characterised uncertainties, physical consistency guarantees, and sufficient interpretability to enable scientific discovery and engineering design. Such models would allow ”rapid screening and idea testing and real-time prediction-based experimental control and optimisation,” enabling scientists and engineers to explore more hypotheses and reach more robust conclusions. See fig. 1.2 for an overview.

For surrogate models to be truly actionable and adopted in industry and scientific research, several critical challenges must be addressed:

Physical Consistency: Machine learned models are often ignorant of fundamental laws of physics and can result in ill-posed problems or non-physical solutions. Physics-informed machine learning approaches, which integrate mathematical physics models with data-driven learning, are essential for ensuring physically consistent predictions.

Uncertainty Quantification: When replacing established physics-based models with surrogate models, quantifying the uncertainty in predictions is crucial. Modelling complex systems requires the inclusion and characterisation of uncertainties and errors that enter at various stages of the computational workflow.

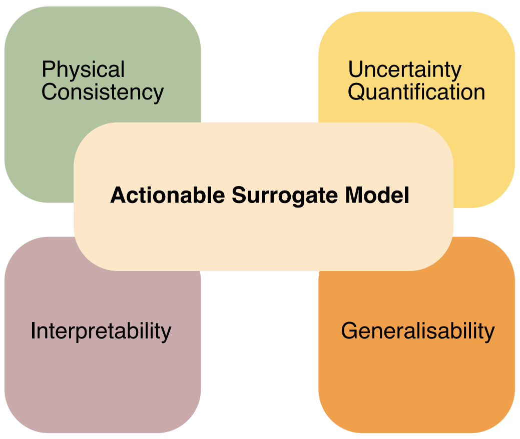

Figure 1.2: Four tenets of an Actionable Surrogate Model.

Interpretability: Black-box surrogate models may achieve impressive performance but lack the interpretability required for scientific understanding and engineering decision-making.

Generalisability: Surrogate models trained on specific scenarios may not generalise well to out-of-distribution scenarios, limiting their broader applicability. Methods that enable surrogate models to extrapolate beyond their training distribution are crucial for real-world deployment. For a neural PDE, generalisability extends to different physics, geometries and boundary conditions.

1.0.4 The Need for Uncertainty Quantification

The challenge of uncertainty quantification (UQ) for neural differential equations is multifaceted. Traditional Bayesian approaches [14], while theoretically sound, often struggle with the high dimensionality and computational demands of complex PDEs. Alternative methods like dropout-based approaches [15] or ensemble techniques [16] may provide uncertainty estimates but lack statistical guarantees. Recent work on conformal prediction offers a promising avenue, providing distribution-free prediction intervals with guaranteed coverage properties [17]. However, extending these methods to high-dimensional spatio-temporal domains presents its own challenges, requiring careful consideration of exchangeability assumptions and computational efficiency.

1.0.5 Combining Experimental and Simulation Data for Scientific Discovery

A persistent challenge in computational science and engineering is the ”sim2real gap”—the discrepancy between simulation predictions and real-world observations. Traditional physics-based simulations, while mathematically rigorous, often rely on simplifying assumptions and idealised conditions that fail to capture the full complexity of physical systems. Actionable surrogate models offer a promising approach to bridge this gap by synergistically combining physics-based knowledge encoded in PDEs with experimental measurements.

By integrating the structured inductive biases of physical equations with the flexibility of data-driven learning, these models can account for phenomena that are difficult to model from first principles alone, such as material imperfections, boundary interactions, and emergent behaviours. In fusion applications, for example, a surrogate model could integrate theoretical plasma physics with experimental diagnostics to better predict anomalous transport phenomena that pure simulation struggles to capture. Similarly, in fluid dynamics, surrogate models can incorporate both Navier-Stokes principles and real-world measurements to account for complex turbulence effects across diverse flow regimes.

This hybrid approach not only improves prediction accuracy but also enhances model generalisability, allowing the surrogate to perform reliably in previously unseen conditions. Moreover, when equipped with proper uncertainty quantification, these models can identify where the sim2real gap is largest, guiding targeted experiments and model refinements in an iterative, scientifically principled manner. The ability to seamlessly assimilate both theoretical knowledge and experimental evidence positions actionable surrogate models as powerful tools for scientific discovery and engineering innovation, enabling more reliable extrapolation from limited data and accelerating the pathway from simulation to real-world application.

1.0.6 Building Digital Twins using Surrogate Models

The development of actionable surrogate models is particularly critical for the creation of digital twins—virtual replicas of physical systems that can reliably simulate their behaviour in real-time. In fusion energy research, digital twins would enable researchers to rapidly explore plasma scenarios, optimise reactor designs, and predict maintenance needs without the enormous costs and delays of physical testing. For instance, a digital twin of a tokamak could predict plasma instabilities before they occur, allowing for preventive control measures that extend operation time and improve performance.

In weather and climate modelling, actionable surrogates can facilitate high-resolution simulations of coastal flooding, extreme weather events, and climate tipping points at speeds compatible with early warning systems and adaptive policy planning [18].

Engineering applications such as aerospace design, materials development, and civil infrastructure benefit similarly—digital twins built with reliable surrogate models can predict structural fatigue, material degradation, and system performance throughout lifecycle phases, allowing for predictive maintenance and design optimisation with quantified confidence levels.

In all these domains, the transformative potential of digital twins can only be realised if the underlying surrogate models not only execute swiftly but also provide physically consistent predictions with well-characterised uncertainty bounds, enabling decision-makers to take appropriate actions based on simulation outcomes they can trust.

1.0.7 Summary

Achieving this vision requires a concerted effort across multiple disciplines, bridging the gap between computational sciences, physics, and engineering. It necessitates not only technical advancements in neural network architectures and uncertainty quantification methods but also careful validation against experimental data and integration with domain knowledge [19]. The path forward involves developing surrogate models that are not just fast approximations but trusted partners in the scientific and engineering design process, capable of providing insights with quantified confidence levels suitable for high-consequence decision-making.

This report addresses the current state of the art and the pressing challenges in surrogate modelling and uncertainty quantification for neural differential equations, focusing on developing methods that maintain physical consistency while providing reliable uncertainty estimates. By advancing the state-of-the-art in this field, we aim to contribute to the broader goal of making computational simulations more accessible, reliable, and actionable across scientific and engineering disciplines, particularly in the journey toward commercial fusion energy, weather modelling and other grand challenges of our time.

1.1 The Objective

1.1.1 PDE modelling

Consider the arbitrary partial differential equation (PDE) modelling the spatio-temporal evolution of n field variables $u \in \mathbb{R}^n$:

$$D = D_t(u) + D_X(u) = 0, \quad X \in \Omega,\, t \in [0,T], \tag{1.1}$$

$$u(X,t) = g(X,t), \quad X \in \partial\Omega\, t \in [0,T], \tag{1.2}$$

$$u(X,0) = a(\lambda, X). \tag{1.3}$$

Here, X defines the spatial domain bounded by Ω and defined with geometry ω. The PDE is defined over a continuous temporal domain $[0,T]$ and D_X and D_t, the composite operators of the associated spatial and temporal derivatives, including the PDE coefficients. The PDE is further defined by the boundary condition $g(X,t)$ and the initial condition $a(\lambda, X)$, parameterised by λ. Our objective resembles the forward problem of solving a PDE setup as an initial-boundary value problem (ibvp). Mathematically, we need to perform the mapping defined below:

$$\mathcal{U} : \mathcal{P} \to \mathcal{U}. \tag{1.4}$$

Here, $\mathcal{P}$ is the space of all possible PDE problems, characterised by the operators D_X, D_t, domain geometry ω, boundary conditions g, and initial conditions a and $\mathcal{U}$ is the space of all PDE solutions.

1.1.2 Neural PDE Modelling

Within the context of neural PDE modelling, we aim to achieve our objective by devising a parameterised neural network $G(\theta)$ to serve as a surrogate model that can approximate the mapping defined within eq. (1.4):

$$\mathcal{N}_\theta \approx \mathcal{U}. \tag{1.5}$$

For a given PDE problem $p \in \mathcal{P}$, the surrogate model generates a prediction of the PDE solution u:

$$\mathcal{N}_\theta(p)(X,t) \approx u(X,t), \tag{1.6}$$

which approximates the true solution as defined in eqs. (1.1) to (1.3).

1.1.3 Error Metrics and Constraints

To evaluate the performance of our surrogate model, we define error metrics that quantify both the accuracy of the approximation and its adherence to the underlying physical principles. The point-wise approximation error can be defined as:

$$\varepsilon(X,t) = ||\mathcal{U}(p)(X,t) - \mathcal{N}_\theta(p)(X,t)||. \tag{1.7}$$

The overall solution accuracy can be measured using integral norms:

$$\mathcal{E}_{\text{sol}} = \left(\int_\Omega \int_0^T ||\mathcal{U}(p)(X,t) - \mathcal{N}_\theta(p)(X,t)||^2 \, dt \, dX \right)^{1/2}. \tag{1.8}$$

Beyond point-wise direct solution accuracy, we can define physics-informed error metrics that measure how well the surrogate model satisfies the governing equations and obeys the conservation laws:

$$\mathcal{E}_{\mathrm{PDE}} = \left(\int_{\Omega} \int_{0}^{T} \| D(\mathcal{N}_{\theta}(p)(X,t)) \|^2 \, dt \, dX \right)^{1/2}. \tag{1.9}$$

Here, D represents the residual operator of the PDE (obtained by expressing eq. (1.1) in it's canonical form). This measures the deviation of the surrogate model prediction from the true physics when deployed to solve for a specific PDE solution. By formulating the PDE in its conservative form, eq. (1.9) becomes a measure of the deviation from the conservation laws for the approximate prediction of the neural PDE.

To ensure the surrogate model remains physically meaningful and sufficiently accurate, we impose the following constraints:

$$\mathcal{E}_{\mathrm{sol}} \leq \varepsilon_{\mathrm{tol}}, \tag{1.10}$$

$$\mathcal{E}_{\mathrm{PDE}} \leq \delta_{\mathrm{tol}}, \tag{1.11}$$

where $\varepsilon_{\mathrm{tol}}$ and δ_{tol} are user-defined, problem-specific tolerance thresholds for solution accuracy and PDE residual, respectively.

1.1.4 Uncertainty Quantification as a Fallback

For neural PDE based surrogate models to become truly actionable in high-stakes scientific and engineering applications, they must navigate a critical balance between error constraints and uncertainty quantification. Ideally, these models should simultaneously satisfy both solution accuracy ($\mathcal{E}_{\mathrm{sol}} \leq \varepsilon_{\mathrm{tol}}$) and physics-informed ($\mathcal{E}_{\mathrm{PDE}} \leq \delta_{\mathrm{tol}}$) error constraints, ensuring predictions that are not only accurate compared to ground truth but also respect underlying physical laws; however, real-world complexity makes strict adherence to it challenging. This is more pronounced, particularly in regions with sparse training data, regime extrapolation, or complex multi-physics interactions. In those situations an actionable surrogate must transition from confident prediction to informed uncertainty quantification, providing calibrated estimates for both solution uncertainty and potential physics violations through spatiotemporal uncertainty maps. This allows us to precisely identify where the model might operate outside its error constraints, thereby enabling selective model refinement, adaptive sampling strategies, and hybrid approaches that blend surrogate predictions with traditional solvers in high-uncertainty regions, ultimately transforming these surrogates from mere computational accelerators to decision-support tools that engineers and scientists can confidently use to guide design exploration, determine when additional high-fidelity simulations are needed. This nuanced approach to error constraints and uncertainty quantification moves beyond the binary notion of either trusting or distrusting surrogate models. Helping make informed judgments for high-consequence decisions across applications from fusion energy research to climate modelling, creating a continuum of reliability that adapts to the specific domain, available data, and physical complexity while maintaining the scientific rigour necessary for addressing our era's grand challenges.

Chapter 2

Background

2.1 Surrogate Modelling

As defined in section 1.1, surrogate modelling of PDEs involves learning to map the spatio-temporal evolution of physical variables within a defined domain. The data under consideration inherits certain implicit biases, such as causality, sequentiality and, depending on the physics under consideration, a range of invariances and equivariances. The surrogate model to serve as the neural PDE needs to be built considering this implicit bias within the data to solve an initial-boundary value problem as defined in eqs. (1.1) to (1.3).

An ideal surrogate model defined in eq. (1.6), should be able to model any PDE(s), explore multiple boundary conditions, provide predictions that are continuous in space and time, and can adapt to any domain geometry all the while being within the tolerance limits as defined in eqs. (1.10) and (1.11). Though the current research landscape is yet to devise a method for constructing an ideal surrogate model, there have been significant developments in providing some of the above-mentioned features to the model [20].

We explore the existing literature under the guise of the physics-data tradeoff that each of these methods espouses. In fig. 2.1, we outline this physics-data axis along which different kinds of surrogate models lie. Along the left end of the axis, the models are physics-driven, relying on less data and more information about the PDE under consideration, in the form of residuals or conservation equations. The further

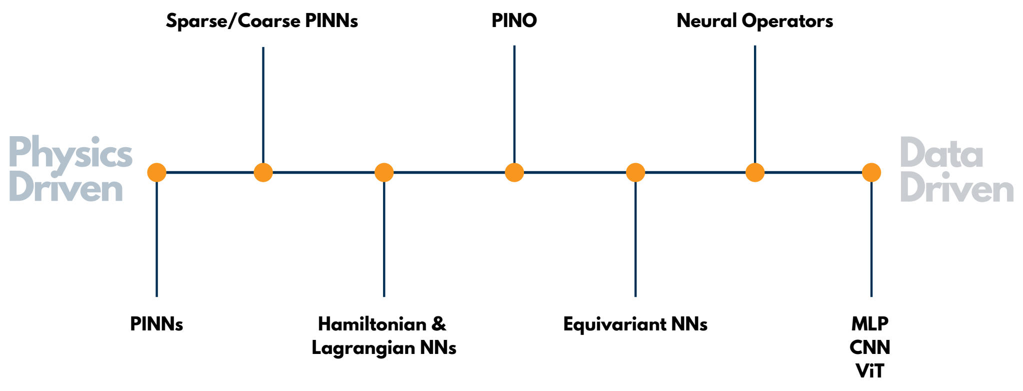

Figure 2.1: Neural PDE landscape: Most surrogate models deployed within SciML frameworks exist along the given Physics-Data axis. On the left end, we have models driven explicitly by physical constraints with limited dependence on data. As we move further right along this axis, the model's explicit knowledge of the physics fades, increasing the dependence on training data from which the model hopes to distill the physics dynamics under consideration.

right we move on the axis, the model becomes more data-driven, implicitly learning the physics of the PDE from simulation/experimental data. Though the primary focus will be on the capability to model the physics well, for each type of neural PDE, we will also be looking at its capability to model and handle **multi-physics**, predict across the **space-time continuum** for desired **geometries** while adapting to different **boundary conditions**.

2.1.1 Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) are a class of machine learning models that incorporate physical laws and domain knowledge directly into the neural network training process through the use of PDE residuals as soft constraints [21].

Given an arbitrary partial differential equation (PDE) system as given in eq. (1.1):

$$D = D_t(u) + D_X(u) = 0, \quad X \in \Omega,\, t \in [0,T]$$

, with boundary conditions $u(X,t) = g(X,t)$ on $X \in \partial\Omega$ and initial conditions

$u(X,0) = a(\lambda,X)$, a PINN represents the solution $u(X,t)$ using a neural network $\mathcal{N}_\theta(X,t)$ with parameters θ. The key innovation of PINNs is the construction of a composite loss function that penalises:

1. The PDE residual: $\mathcal{L}_{\text{PDE}} = \|D_t(\mathcal{N}_\theta) + D_X(\mathcal{N}_\theta)\|^2_{\Omega\times[0,T]}$
2. The boundary condition mismatch: $\mathcal{L}_{\text{BC}} = \|\mathcal{N}_\theta - g\|^2_{\partial\Omega\times[0,T]}$
3. The initial condition mismatch: $\mathcal{L}_{\text{IC}} = \|\mathcal{N}_\theta(\cdot,0) - a(\lambda,\cdot)\|^2_\Omega$

The total loss is a weighted sum of the PDE residual, the error with the initial and boundary conditions:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}}, \tag{2.1}$$

leading to an unsupervised training objective:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta). \tag{2.2}$$

This approach effectively embeds the physics described by the PDE system directly into the neural network training process. Traditionally, PINNs are devised as coordinate-based MLPs (implicit neural representations) that learn to map the space-time coordinates to physical field variables [22, 23]. PINNs serve as surrogates that approximate the mapping $\mathcal{U}$ from the PDE problem space to the solution space while inherently respecting the conservation laws encoded in the PDE formulation. Unlike traditional numerical methods, PINNs leverage automatic differentiation to evaluate the PDE residuals, eliminating the need for mesh generation and making them particularly advantageous for high-dimensional problems or complex geometries. PINNs have found application in a range of modelling tasks, ranging from fluid mechanics [24], molecular and material deformation [25] to magnetic control of fusion devices [26]. There exist various variants to PINNs that rely on modelling using a physics-driven approach. Deep Galerkin methods minimise a volumetric residual applied over a test function [27], Deep Ritz method where the loss is defined as the energy of the problem's solution and minimised [28]. PINNs are also structured to solve the conservative form of the PDE [29], allowing for better representations of sources and sinks.

Limitations: Though PINNs mathematically guarantee convergence to the solution, in practice, they often fail to meet this criterion. They suffer from the errors arising from the approximation, generalisation and optimisation found within neural networks [30]. Faced with a multi-objective optimisation task as given in eq. (2.1), they fail to satisfy all objectives, imperative for a well-defined PDE [31]. They are often ill-conditioned, leading to a poorly defined loss landscape unable for gradient descent-based algorithms to traverse and identify a suitable minima [32].

Geometry: Being inherently mesh-free in design, PINNs are capable of adapting to different geometries within the same spatio-temporal problem domain. However, due to the lack of an explicit definition of the domain geometry, they fail to model the boundary conditions imposed by complex geometries and need additional geometry information as inputs to aid the modelling [33]. **Space-time Continuum:** The mesh-free nature allows PINNs to be continuous in space and time, generating predictions that are smooth, infinitely differentiable solutions across the interested domain. **Multi-Physics:** PINNs generally struggle with complex multiscale and multiphysics settings as it adds more objectives to the loss function, increasing the complexity of the optimisation task in hand. **Boundary Conditions**: PINNs either encode the boundary condition as a soft-constraint in the loss or as a hard-constraint within the architecture [34, 35], however they remain fixed after training and fail to adapt to other boundary conditions when deployed.

2.1.1.1 Sparse/Coarse PINNs

Considering the challenge of optimisation for PINNs, when trained in an unsupervised setting, various works have proposed the utilisation of limited simulation/experimetnal data to help augment the loss landscape [30, 36]. In these works, the loss in eq. (2.1) is modified to have a data element as well:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{Datas}}\mathcal{L}_{\text{Data}}, \tag{2.3}$$

where $\mathcal{L}_{\text{Data}}$ is the L2 norm taken across the model predictions and the target data. The target data could be obtained from sparse experimental data or simulation data generated on a coarser grid. Supplying data to the training objective transforms the optimisation from an unsupervised learning scenario to a hybrid supervised-unsupervised model, allowing the model to approximate the solution better. However, the optimal choice of the coefficients λ within the loss functions still remains an open problem [37].

2.1.1.2 Lagrangian and Hamiltonian Neural Networks

Lagrangian and Hamiltonian Neural Networks are specialised machine learning architectures that incorporate fundamental principles from classical mechanics to model physical systems with improved accuracy and physical consistency.

Lagrangian Neural Networks (LNNs) learn to approximate the Lagrangian function ($L = T - V$, where T is kinetic energy and V is potential energy) of a physical system [38]. By learning this scalar function and applying the Euler-Lagrange equations through automatic differentiation, LNNs can predict system dynamics while naturally preserving conservation laws and symmetries. This approach is particularly effective for systems where conservation of energy and momentum are essential.

LNNs learn the Lagrangian function and use the Euler-Lagrange equations:

$$\mathcal{L}_{\text{LNN}} = \frac{1}{N}\sum_{i=1}^{N}\left(\left\|\ddot{q}_i - \frac{d}{dt}\left(\frac{\partial L_\theta}{\partial \dot{q}}\right)_i + \left(\frac{\partial L_\theta}{\partial q}\right)_i\right\|^2\right), \tag{2.4}$$

where $L_\theta(q, \dot{q})$ is the neural network approximation of the Lagrangian with parameters θ, q represents generalized coordinates and $\dot{q}$ their time derivatives. The Euler-Lagrange equation $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = 0$ governs the dynamics

Hamiltonian Neural Networks (HNNs) instead learn the Hamiltonian function $(H = T + V)$ of a system. By learning this energy function and implementing Hamilton's equations through automatic differentiation, HNNs explicitly preserve the symplectic structure of physical systems. This makes them exceptionally well-suited for modelling conservative dynamical systems and providing stable long-term predictions [39]. Both approaches offer significant advantages over conventional neural networks by embedding physical inductive biases directly into their architecture, resulting in models that require less training data and produce physically consistent predictions that respect fundamental conservation laws.

HNNs learn the Hamiltonian function and use Hamilton's equations:

$$\mathcal{L}_{\text{HNN}} = \frac{1}{N}\sum_{i=1}^{N}\left(\left\|\dot{q}_i - \left(\frac{\partial H_\theta}{\partial p}\right)_i\right\|^2 + \left\|\dot{p}_i + \left(\frac{\partial H_\theta}{\partial q}\right)_i\right\|^2\right) \tag{2.5}$$

Where: $H_\theta(q,p)$ is the neural network approximation of the Hamiltonian with parameters θ, q represents generalized coordinates and p their conjugate momenta. Hamilton's equations $\dot{q} = \frac{\partial H}{\partial p}$ and $\dot{p} = -\frac{\partial H}{\partial q}$ govern the dynamics

Both LNNs and HNNs can incorporate additional regularization terms or constraints for specific physical properties as needed.

2.1.2 Equivariant Neural Networks

Equivariant Neural Networks (EqNNs) are a class of neural architectures designed to respect the underlying symmetries and invariances present in physical systems. By embedding these symmetries directly into the network architecture, EqNNs can provide more physically consistent solutions with improved sample efficiency compared to standard neural networks [40, 41, 42, 43].

Let G be a symmetry group acting on the input space $\mathcal{X}$ and output space $\mathcal{Y}$ via group actions $\rho_{\mathcal{X}}$ and $\rho_{\mathcal{Y}}$, respectively. A function $f : \mathcal{X} \to \mathcal{Y}$ is said to be equivariant with respect to G if for all $g \in G$ and $x \in \mathcal{X}$:

$$f(\rho_{\mathcal{X}}(g)x) = \rho_{\mathcal{Y}}(g)f(x) \tag{2.6}$$

In the context of PDEs as described in eq. (1.1), the equivariance property ensures that if the input data (e.g., initial conditions, boundary conditions) undergoes a transformation from group G, the predicted solution transforms in a corresponding manner. For instance, if we rotate the boundary conditions of a fluid dynamics problem, an equivariant neural network would produce a solution that is exactly the rotated version of the solution for the original boundary conditions.

Equivariant neural networks employ specialised architectural components that preserve group symmetries throughout the forward pass. Two main approaches have emerged for integrating symmetries into neural architectures [42]:

1. **Group Convolutional Networks**: These generalize standard convolutions to operate over a symmetry group G, where the convolution operation naturally preserves equivariance [41]. Group convolution is defined for a compact group G with kernel $K : G \to \mathcal{L}(V_1, V_2)$, feature function $f : G \to V_1$, and Haar measure μ on G as:

$$(K * f)(s) = \int_G K(r^{-1}s)f(r)d\mu(r) \tag{2.7}$$

   This operation can be extended to representative group convolutions that support non-regular representations, providing greater flexibility in modelling various symmetries [42].

2. **Steerable Convolutional Networks**: These leverage steerable filters that transform predictably under group actions [44]. Tensor Field Networks [45] extend this approach to operate on geometric tensor fields to maintain SE(3) equivariance.

For solving PDEs, the loss function for an equivariant neural network $\mathcal{E}_\theta$ could combine the standard PDE loss terms with equivariance constraints:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{IC}} + \lambda_{\text{eq}} \mathcal{L}_{\text{eq}} \tag{2.8}$$

$$\mathcal{L}_{\text{eq}} = \mathbb{E}_{g \in G, x \in \mathcal{X}} \, \| \mathcal{E}_\theta(\rho_{\mathcal{X}}(g)x) - \rho_{\mathcal{Y}}(g) \mathcal{E}_\theta(x) \|^2 \tag{2.9}$$

where λ_{eq} balances the equivariance constraint against other loss terms. However, in practice, when the architecture is perfectly equivariant by construction, $\mathcal{L}_{\text{eq}}$ becomes unnecessary, as equivariance is guaranteed for any parameter values θ [40, 42].

A particularly powerful approach for solving PDEs with equivariant networks involves leveraging the theory of differential invariants [42]. For a PDE system with symmetry group G, the differential operators can be expressed in terms of differential invariants:

$$\sum_{i=1}^{k_t} a_i \partial_t^i u = F\left(\partial \phi_{u,n}^{G,1}, \ldots, \partial \phi_{u,n}^{G,k} \right) \tag{2.10}$$

where $\partial \phi_{u,n}^{G,j}$ are the differential invariants of the group G. Equivariant neural networks can be trained to approximate these invariants, enabling the construction of symmetry-preserving discretisation schemes without the need to formally derive numerical invariantizations [42].

Equivariant neural networks have demonstrated success in modelling various physical systems governed by PDEs. They have been used for turbulence modelling with symmetry preservation [46]. E(3)-equivariant mesh-based networks have shown superior performance in predicting turbulent flows by preserving rotational and translational symmetries [40]. Group-convolutional approaches have been successfully applied to 2D heat equations through SE(2) symmetry-preserving discretisations [42]. Spherical CNN architectures that respect the symmetries of the sphere have been applied to global weather prediction tasks [47, 48]. SE(3)-equivariant networks have been used to predict the molecular wave functions of atomistic systems, guaranteeing rotational and translational invariance of energy predictions [49]. Exploiting Lie point symmetries within the training data allows for further data augmentation, reducing sample complexity by an order of magnitude while maintaining physical consistency [50]. One of the most successful approaches is E(n)-Equivariant Graph Neural Networks [51], which combine the expressivity of graph neural networks with equivariance to Euclidean transformations, proving particularly effective for modelling particle systems and continuum mechanics.

Limitations: Despite their theoretical advantages, equivariant neural networks face several challenges. They often require specialised implementations that can be computationally intensive, particularly for higher-dimensional groups or complex symmetries [52]. The formal derivation of numerical invariantization becomes increasingly challenging as the number of variables increases, limiting the applicability of these methods in high-dimensional settings [42]. The strict enforcement of symmetries may also limit the network's expressivity in cases where the underlying system exhibits near-symmetries or broken symmetries [53].

Geometry: Equivariant neural networks excel in handling complex geometries as they can naturally incorporate geometric transformations. For instance, SE(3)-equivariant networks can model rotationally invariant phenomena regardless of the orientation of the domain [45]. However, they may struggle with domains that have symmetry-breaking boundary features unless specifically designed to accommodate them [40]. For PDEs defined on non-Euclidean manifolds, specialised equivariant architectures have been developed to respect the underlying geometric structure [42].
Space-time Continuum: Depending on the architecture being deployed, equivariant networks can operate on continuous space-time domains or on fixed discrete domains. They often leverage positional encodings that respect the underlying symmetries of the space-time manifold [23]. **Multi-Physics:** Equivariant neural networks are particularly well-suited for multi-physics problems where different physical phenomena share underlying symmetries. By enforcing these symmetries in the architecture, they can more efficiently learn the complex interactions between different physi-

cal processes[43]. An advantage is that once an equivariant architecture is trained to approximate the differential invariants of a specific symmetry group, it can be applied to solve any other PDE with the same symmetry group without retraining [42]. **Boundary Conditions:** Enforcing boundary conditions while maintaining equivariance can be challenging, particularly for Dirichlet or Neumann conditions that may break certain symmetries. Various approaches have been proposed, including symmetry-preserving boundary constraint functions and augmented Lagrangian methods that incorporate boundary conditions as soft constraints while preserving the equivariant structure of the network [53]. For symmetry-preserving finite difference schemes, the formulation of boundary conditions in terms of differential invariants allows for the construction of discretisation schemes that respect both the PDEs and their boundary conditions [42].

2.1.3 Neural Operators

Neural Operators are a class of machine learning models that learns to approximate the operator that maps between function spaces from a finite collection of observed input-output pairs [54]. Given input functions from a Banach space $\mathcal{A}$ and output solutions in a Banach space $\mathcal{U}$, a neural operator learns the mapping:

$$NO_\theta : \mathcal{A} \to \mathcal{U} \tag{2.11}$$

The core notion behind a neural operator is that it depends on discretisation-invariant functions rather than on discretised vectors. This is achieved by constructing the models to implement a method of kernel integration that takes the shape:

$$\big(\kappa(a;\phi)v_t\big)(x) = \int_D \kappa\big(x, y, a(x), a(y); \phi\big) v_t(y) dy, \tag{2.12}$$

The neural operator defined in eq. (2.12) represents the non-local transformation of hidden representation v_t to produce values at position x. Here, κ is a learnable kernel parameterised by ϕ, allowing the model to capture relationships between input function values $a(x)$ and $a(y)$ across the entire domain D. This kernel integration appears within the iterative layer update:

$$v_{t+1}(x) = \sigma\left(W_t v_t(x) + \left(\kappa(a;\phi)v_t\right)(x) + b_t(x)\right), \tag{2.13}$$

where σ is a nonlinear activation function, W_t represents a local linear transformation capturing point-wise features, and $b_t(x)$ is a bias term. The complete neural operator architecture consists of an initial lifting operator P mapping input function $a(x) \mapsto v_0(x)$, followed by T layers of these kernel integrations with nonlinearities, and concluding with a projection operator Q mapping $v_T(x) \mapsto u(x)$ to produce the output function.

There exist different implementations of neural operators based on the method of kernel integration, each with its own set of advantages and disadvantages as outlined in table 2.1.

Neural Operators have found application in modelling complex PDE across a range of domains from weather modelling [61, 62, 63], carbon monitoring [64, 65], fluid modelling [66, 67] and nuclear fusion [68, 13, 69, 70]. Neural Operators have been devised to formulate foundation physics models trained on a large PDE dataset with the aim to generalise to unseen physics [71, 72, 73].

Limitations: Neural operators are only discretisation invariant in theory. In practice, because they are trained using discretised instances of the input functions and output solutions, they are rather discretisation-convergent [74, 75]. The extent of this convergence depends on the method of kernel integration deployed and on the aliasing errors that arise from the discretisation of the training data. Multigrid methods have demonstrated potential in addressing these aliasing errors [76]. Depending on the choice of kernel, these models can be relatively complex in operation and have a large memory footprint as well. **Space-time Continuum:** As described above, neural operators allow for continuous predictions in the spatial domain, but are discrete in the temporal dimension as they are handled autoregressively [77, 78]. There have been some recent attempts to make them continuous in time, but they have been limited to toy experiments [79]. **Geometry:** Certain neural oeprators are

Neural Operator	**Kernel Integration Method**	**Advantages**	**Disadvantages**
Graph Neural Operator (GNO) [55]	Nyström approximation with domain truncation	• Handles unstructured grids • Flexible domain geometry • Captures local structure efficiently	• Performance depends on sampling quality • Truncation may lose long-range dependencies
Low-rank Neural Operator (LNO) [54]	Tensor product decomposition	• Linear complexity • Efficient for near-linear operators • Simple implementation	• Limited expressivity • Struggles with highly non-linear operators • Rank must be chosen carefully
Multipole Graph Neural Operator (MGNO) [56]	Multi-scale decomposition with hierarchical matrices	• Linear complexity • Captures multi-scale interactions • Works well on complex geometries	• Complex implementation • Hierarchical structure design is problem-dependent • Higher overhead than other methods
Fourier Neural Operator (FNO) [11]	Spectral convolution using FFT	• Fast computation • Excellent for smooth functions • State-of-the-art performance on many PDEs	• Limited to structured grids • Struggles with discontinuities • Periodic boundary conditions assumed
DeepONet [57]	Branch-trunk architecture with point-wise evaluation	• Universal approximation guarantees • Can query at arbitrary output points • Theoretically well-founded	• Not discretisation-invariant in basic form • Input function must be on fixed grid points • Linear approximation scheme
Implicit Neural Representation (INR) [23]	Point-wise MLP with periodic activations	• Continuous representation • Memory efficient • Resolution-independent	• Not an operator (represents single function) • Slow training convergence • Struggles with high-frequency details
Convolutional Neural Operator (CNO) [58]	Bandlimited convolution and continuous-discrete equivalence	• Resolution invariant • Robust multi-scale processing • Excellent generalisation capability • Representation equivalence guarantees	• Requires bandlimited function spaces • Higher implementation complexity • More expensive sinc filtering operations
Laplace Neural Operator (LNO) [59]	Pole-residue Laplace domain transformation	• Captures transient responses • Handles non-periodic signals • Physically interpretable parameters • Exponential convergence capabilities	• Limited to specific applications • Complex parameter interpretation • More suitable for systems with poles and residues
Wavelet Neural Operator (WNO) [60]	Wavelet transform and kernel integration	• Superior time-frequency localisation • Handles discontinuities effectively • Captures spatial patterns well • Works for complex boundaries	• Choice of wavelet basis affects performance • Computational overhead for wavelet transform • More complex implementation

Table 2.1: Comparison of different neural operator architectures based on the kernel integration method, exploring the pros and cons of each.

capable of handling irregular meshes and point cloud data [55, 23, 57], whereas most others are restricted to regular structured grids [11, 58, 59, 60]. **Multi-physics:** Most neural operator usage has been for modelling a specific physics under consideration and they have failed to extend to a more generalisable understanding of the dynamics. **Boundary Conditions:** Within models like the FNO, the boundary conditions are expected to be periodic in nature. Other models operating on structured grids have the capability of handling different boundary conditions by modifying the type of convolution being deployed. These models do not exhibit any distinction of the PDE from its boundary conditions and once trained the boundary conditions are embedded to the dynamics distilled into the model, that they cannot be used to model the physics under different boundary conditions.

2.1.3.1 Physics-Informed Neural Operators

Physics-Informed Neural Operators (PINO) are a learning method for neural PDEs combining a neural operator architecture with the training objective of a PINN as outlined in eq. (2.1) [67]. They are often used in a setting where a neural operator model trained on a general subset of simulation data is further finetuned to solve for a specific, well-defined PDE [80, 81] using the PDE loss alone. PINO models rely on using automatic differentiation or finite differences to evaluate the PDE residuals across the prediction [82]. In this work, message passing within a graph neural operator has been modified to behave as a finite volume method for modelling the dynamics of fluid flow [83].

Composite neural operator models, constructed by combining several neural operator architectures, help bring the best out of each model as seen in this modified implicit neural representation [63]. By combining a GNO with an FNO, the model can handle irregular geometry and provides for efficient kernel integration [84].

2.1.4 Data-Driven Neural Networks

Since the fundamental value of a surrogate model is to provide a quick, inexpensive approximation of the true function, traditional neural network architectures have a significant role to play in the landscape of neural PDE solvers. Through this part, we quickly touch upon fundamental machine learning models, their mathematical formulation, applications for PDE modelling and their limitations

2.1.4.1 Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP) is one of the foundational neural network architectures, consisting of at least three layers: an input layer, one or more hidden layers, and an output layer [85]. Each layer comprises multiple neurons (or nodes), with each neuron in a layer fully connected to all neurons in the subsequent layer. MLPs are feed-forward networks, meaning information flows only in one direction—from input to output—with no cycles or loops.

Given an input vector $\mathbf{x} \in \mathbb{R}^{d_0}$, an MLP with L layers computes:

$$\mathbf{h}_0 = \mathbf{x} \tag{2.14}$$

$$\mathbf{h}_\ell = \sigma_\ell(\mathbf{W}_\ell \mathbf{h}_{\ell-1} + \mathbf{b}_\ell), \quad \text{for } \ell = 1, 2, \ldots, L \tag{2.15}$$

where $\mathbf{h}_\ell \in \mathbb{R}^{d_\ell}$ is the output of the ℓ-th layer, $\mathbf{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ is the weight matrix for ℓ-th layer $\mathbf{b}_\ell \in \mathbb{R}^{d_\ell}$ is the bias vector for the ℓ-th layer σ_ℓ is a non-linear activation function (e.g., ReLU, sigmoid, tanh). The final output $\mathbf{y} = \mathbf{h}_L$ is used for prediction or classification. The network parameters $\theta = \{(\mathbf{W}_\ell, \mathbf{b}_\ell)\}_{\ell=1}^{L}$ are learned by minimizing a loss function $\mathcal{L}(\mathbf{y}, \mathbf{y}^*)$ using gradient-based optimization methods, where $\mathbf{y}^*$ represents the ground truth labels. MLPs have been used as surrogates for modelling wind turbine blades [86], classifiers for high-energy physics [87], and for designing fusion reactors [88].

Limitations MLPs suffer from the curse of dimensionality, requiring exponentially more data as input dimensionality increases. They lack inherent mechanisms to handle sequential or spatial data structures, ignoring important local relationships in data like images or text. Despite their universal approximation properties, MLPs often require significant tuning of hyperparameters and are prone to overfitting on small datasets.

2.1.4.2 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are specialised neural architectures designed to process sequential data by maintaining an internal state (memory) that captures information from previous inputs. Unlike feed-forward networks, RNNs include feedback connections, allowing information to persist across time steps [89]. This recursive structure enables RNNs to model temporal dependencies in data, making them particularly suitable for tasks involving sequences of variable length.

For an input sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$ where each $\mathbf{x}_t \in \mathbb{R}^d$, a simple RNN computes:

$$\mathbf{h}_t = \sigma_h(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.16}$$

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_{yh}\mathbf{h}_t + \mathbf{b}_y) \tag{2.17}$$

where, $\mathbf{h}_t \in \mathbb{R}^m$ is the hidden state at time step t, $\mathbf{y}_t \in \mathbb{R}^n$ is the output at time step t, $\mathbf{W}_{hx} \in \mathbb{R}^{m \times d}$ is the input-to-hidden weight matrix, $\mathbf{W}_{hh} \in \mathbb{R}^{m \times m}$ is the hidden-to-hidden weight matrix, $\mathbf{W}_{yh} \in \mathbb{R}^{n \times m}$ is the hidden-to-output weight matrix, $\mathbf{b}_h \in \mathbb{R}^m$ and $\mathbf{b}_y \in \mathbb{R}^n$ are bias vectors, σ_h and σ_y are activation functions

More advanced RNN variants include Long Short-Term Memory (LSTM) networks [90] and Gated Recurrent Units (GRUs) [91], which introduce gating mechanisms to better control information flow and mitigate the vanishing gradient problem. RNNs and its variants have been used to better model the temporal dependencies of PDEs across a wide range of physical dynamical systems [92, 93, 94]

Limitations: Standard RNNs suffer from the vanishing and exploding gradient problems, making them challenging to train on long sequences. Despite advancements like LSTMs and GRUs, RNNs still struggle to capture very long-range dependencies effectively. The sequential nature of RNN computation prevents parallelisation during training, resulting in slower convergence compared to architectures that allow for more parallel processing.

2.1.4.3 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are specialised neural architectures primarily designed for processing grid-like data, such as images. They incorporate three key principles: local receptive fields, shared weights, and spatial or temporal subsampling [95]. By leveraging these principles, CNNs automatically learn hierarchical representations of data, capturing low-level features (e.g., edges, textures) in early layers and more abstract, high-level features in deeper layers. The convolutional nature of these networks introduces translation invariance, making them robust to spatial shifts in the input data.

For a 2D input $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ (height, width, channels), a convolutional layer computes:

$$\mathbf{Z}_{i,j,k} = \sum_{m=0}^{F_H-1} \sum_{n=0}^{F_W-1} \sum_{c=0}^{C-1} \mathbf{X}_{i\cdot s+m, j\cdot s+n, c} \cdot \mathbf{K}_{m,n,c,k} + \mathbf{b}_k \tag{2.18}$$

$$\mathbf{A}_{i,j,k} = \sigma(\mathbf{Z}_{i,j,k}) \tag{2.19}$$

where, $\mathbf{Z}_{i,j,k}$ is the pre-activation value at position (i, j) for the k-th feature map, $\mathbf{A}_{i,j,k}$ is the activation value after applying non-linearity σ, $\mathbf{K} \in \mathbb{R}^{F_H \times F_W \times C \times K}$ represents K convolutional filters of size $F_H \times F_W \times C$, $\mathbf{b} \in \mathbb{R}^K$ is a bias vector, s is the stride parameter

A typical CNN architecture also includes pooling layers for downsampling:

$$\mathbf{P}_{i,j,k} = \text{pool}(\{\mathbf{A}_{i\cdot p+m, j\cdot p+n, k} \mid 0 \leq m, n < P\}) \tag{2.20}$$

where P is the pool size and p is the pooling stride. Common pooling operations include max-pooling and average-pooling. The final layers of a CNN often consist of fully connected layers for classification or regression tasks, however U-Nets with a fully convolutional network have found tremendous applications in spatio-temporal modelling [96]. They have been used for modelling weather complex PDEs [97], nuclear fusion [98] and even to mimic a multigrid method for fluid modelling [99].

Limitations: CNNs lack explicit mechanisms to model long-range dependencies in data, limiting their effectiveness on tasks requiring global context understanding. They typically require large amounts of labelled data for effective training, making them less suitable for domains with limited annotated examples. Despite their translation invariance, CNNs are not inherently invariant to other transformations such as rotation, scaling, or viewpoint changes without specific architectural modifications or data augmentation.

2.1.4.4 Vision Transformer (ViT)

The Vision Transformer (ViT) adapts the Transformer architecture, originally designed for natural language processing, to computer vision tasks. Unlike CNNs, which use convolution operations to process images, ViT treats an image as a sequence of patches and processes them using self-attention mechanisms [100]. This approach allows the model to capture global dependencies in the data without the inductive biases inherent in CNNs.

Given an input image $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, ViT first divides it into N non-overlapping patches $\mathbf{x}_p \in \mathbb{R}^{P^2 \cdot C}$, where (P, P) is the patch size. These patches are linearly projected to obtain patch embeddings:

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{E}\mathbf{x}_1; \mathbf{E}\mathbf{x}_2; \ldots; \mathbf{E}\mathbf{x}_N] + \mathbf{E}_{\text{pos}} \tag{2.21}$$

where, $\mathbf{x}_{\text{class}}$ is a learnable classification token prepended to the sequence, $\mathbf{E} \in \mathbb{R}^{D \times (P^2 \cdot C)}$ is a learnable linear projection, $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$ contains positional embeddings.

The Transformer encoder processes these embeddings through L layers of multi-head self-attention (MSA) and MLP blocks:

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \ldots L \tag{2.22}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \quad \ell = 1 \ldots L \tag{2.23}$$

where LN denotes layer normalization. The self-attention mechanism is defined as:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.24}$$

where **Q**, **K**, and **V** are query, key, and value matrices. The multi-head attention extends this by projecting the queries, keys, and values h times with different learned projections.Transformers have found application in PDE solving [101, 71, 102] and weather modelling [103].

Limitations Vision Transformers require large datasets for training from scratch, often performing worse than CNNs when trained on smaller datasets without strong regularization. The quadratic complexity of self-attention mechanisms with respect to sequence length makes ViTs computationally expensive for high-resolution images. Despite their strong performance, ViTs lack some of the useful inductive biases of CNNs, such as translation equivariance and locality.

2.2 Uncertainty Quantification

Though uncertainty quantification for neural networks is a thesis by itself, we focus primarily on three broad categories that find consistent application or are emerging methods within the field of scientific machine learning. As given in fig. 2.2, we categorise UQ methods into three categories: Bayesian, frequentist and verified methods. Within each category, we focus on three submethods that find application for modelling PDEs and other dynamical systems. The key features, advantages and disadvantages of each method are outlined in table 2.2. We dive deeper into the formulation, application and limitations of each of the methods in the sections below.

2.2.1 Bayesian Methods

2.2.1.1 Bayesian Neural Networks (BNNs)

Bayesian Neural Networks (BNNs) extend traditional neural networks by treating weights as probability distributions rather than point estimates, providing a principled framework for uncertainty quantification in regression tasks. Unlike deterministic networks that output a single predicted value for each input, BNNs produce predic-

UQ Method	**Key Features**	**Advantages**	**Disadvantages**
		Bayesian Methods	
Bayesian Neural Networks [104]	Treats weights as random variables • Learns posterior distribution • Various inference techniques (VI, HMC)	Principled uncertainty representation • Captures aleatoric and epistemic uncertainty • Natural incorporation of priors	High computational cost • Challenging to scale • Often requires approximations • Fails to guarantee coverage
Monte Carlo Dropout [15]	Dropout during inference • Multiple forward passes • Approximates VI in infinite width networks	Simple to implement • Works with existing architectures • Minimal training overhead	Limited expressiveness • Dropout rate affects quality • Often underestimates uncertainty
Deep Ensembles [16]	Multiple networks with different initializations • Can use adversarial training • Combined at inference	• Captures multi-modal uncertainties • Robust to distribution shift	High training cost • Memory intensive • Multiple model maintenance • Noisy and fails to guarantee coverage
		Frequentist Methods	
Evidential Networks [105]	Predicts parameters of conjugate priors • Second-order uncertainty • End-to-end training	Single forward pass at inference • Uncertainty in uncertainty • Aleatoric/epistemic separation	Constrained by prior family • Potential overconfidence • Careful loss design needed
Conformal Prediction [106]	Distribution-free intervals • Guarantees coverage probability • Can be applied post-hoc	Rigorous statistical guarantees • Model-agnostic • Calibrates any prediction model	Requires calibration set • May produce wide intervals • Limited for OOD data
Bootstrap Ensembles [107]	Resamples training data • Training on different subsets • Frequentist posterior analog	No distributional assumptions • Simple implementation • Parallelizable training	Computationally expensive • Data inefficient • May underestimate uncertainty
		Verified Methods	
Interval Arithmetic [108]	Propagates interval bounds • Guaranteed to contain true output • Maintains bounds per operation	Mathematically rigorous • Complete verification • Conceptually simple	Often produces loose bounds • Dependency problem • Limited scalability
Taylor Models [109]	Polynomial approximation with error bounds • Higher-order tracking	Tighter bounds than intervals • Captures dependencies • Handles nonlinearities well	Complex implementation • Expensive for higher order • Inefficient for deep networks
Zonotopes [110]	Linear transformations of hypercubes • Affine set representation • Specialized for ReLU networks	Formal guarantees with rigorous bounds • Exact for affine layers • Capture non-convex output distributions	Approximation for nonlinearities • Memory grows with depth • Computing exact bounds expensive for high dimensions

Table 2.2: Comparison of Uncertainty Quantification Methods for Neural Networks

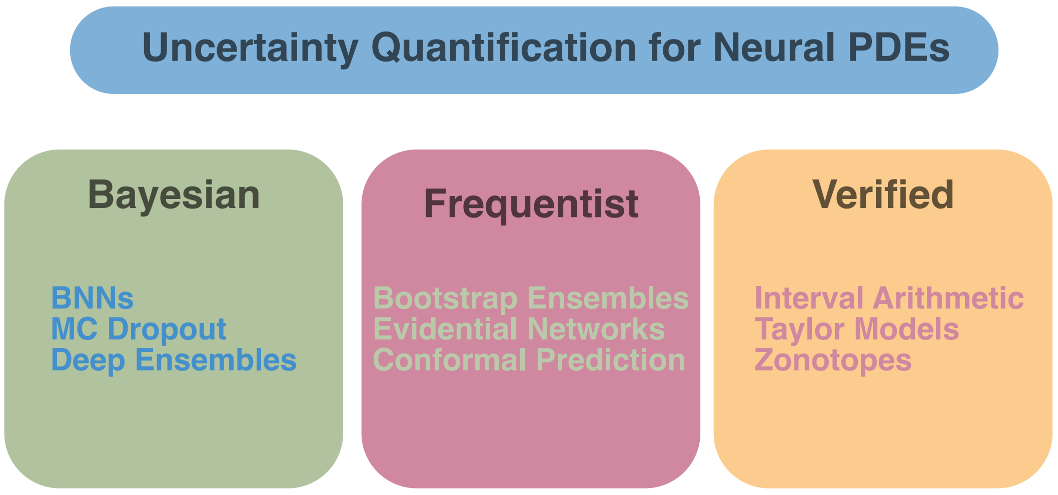

Figure 2.2: High-level landscape of various UQ methods available for Neural PDEs

tive distributions that capture both the expected value and the associated uncertainty, enabling more robust predictions in regression problems where quantifying confidence is crucial, such as in scientific and engineering applications where safety considerations or decision-making under uncertainty are paramount.

In standard regression neural networks, weights **w** are optimised as point estimates to minimise a loss function (typically mean squared error):

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathcal{D}) = \min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 \tag{2.25}$$

where $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ represents the training data, and $f(\mathbf{x}; \mathbf{w})$ is the neural network function. BNNs, however, place a prior distribution $p(\mathbf{w})$ over the weights (often Gaussian) and learn a posterior distribution $p(\mathbf{w}|\mathcal{D})$ via Bayes' rule:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \tag{2.26}$$

For regression, the likelihood $p(\mathcal{D}|\mathbf{w})$ is typically modeled as Gaussian:

$$p(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(y_i | f(\mathbf{x}_i; \mathbf{w}), \sigma^2) \tag{2.27}$$

where σ^2 represents the observation noise variance. Predictions for new inputs $\mathbf{x}^*$ are made by marginalising over the posterior to obtain a predictive distribution:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w} \tag{2.28}$$

This integral is typically approximated either via variational inference (by minimising the KL divergence between an approximate distribution $q_\theta(\mathbf{w})$ and the true posterior) or using Monte Carlo methods to draw samples from the posterior.

In scientific regression problems, BNNs have been successfully employed for uncertainty-aware surrogate modelling of complex physical systems, providing probabilistic approximations to solutions of partial differential equations with quantified confidence bounds [111, 112, 19]. BNNs have been deployed to provide UQ for PINNs [113], Physics-informed priors have allowed for better posterior estimation in BNNs [114] and have found utility in PDE discovery [115]. Multifidelity BNNs have found success in modelling PDEs in data-constraint settings [116].

Limitations: Computational complexity remains a significant challenge, as training BNNs through variational inference or MCMC methods requires substantially more computation than their deterministic counterparts, with this cost scaling poorly for deep architectures with millions of parameters. The specification of appropriate priors over weights significantly impacts the quality of uncertainty estimates but lacks principled guidelines beyond conjugate Gaussian priors, which may not adequately capture the true weight distribution in complex neural networks. Furthermore, posterior approximation methods introduce additional limitations: variational inference tends to underestimate predictive uncertainty due to its mode-seeking behavior, while more accurate MCMC methods struggle with convergence in high-dimensional parameter spaces typical of neural networks. Finally, even well-trained BNNs often exhibit calibration issues in regression tasks, where predictive distributions may be systematically too narrow or too wide relative to the true data distribution, requiring additional post-processing calibration techniques to obtain reliable uncertainty estimates for critical applications.

2.2.1.2 Monte Carlo Dropout

Monte Carlo (MC) Dropout represents a practical approximation to Bayesian inference in neural networks, reinterpreting the common regularisation technique of dropout as a variational Bayesian approximation when applied during both training and inference. This approach enables uncertainty quantification in regression tasks without the full computational burden of traditional Bayesian methods, leveraging the insight that a neural network with dropout applied at test time can be viewed as an ensemble of thinned networks sharing parameters, thus providing a computationally efficient way to approximate prediction uncertainty through the statistical properties of multiple stochastic forward passes.

In standard dropout training, network units are randomly masked during training with probability p to prevent co-adaptation and improve generalisation:

$$\hat{\mathbf{y}} = f(\mathbf{x}; \mathbf{W} \odot \mathbf{z}) \tag{2.29}$$

where $\mathbf{z}$ is a binary mask with elements drawn from a Bernoulli distribution with probability $1-p$ of being 1, and $\odot$ represents element-wise multiplication. It was shown that applying dropout at test time and performing multiple forward passes can be interpreted as an approximation to variational inference in a Bayesian neural network with a specific variational distribution $q_\theta(\mathbf{W})$ over the weights [15]:

$$q_\theta(\mathbf{W}_i) = \mathbf{M}_i \cdot \mathrm{diag}([z_{i,j}]_{j=1}^{K_i}) \tag{2.30}$$

where $\mathbf{M}_i$ are variational parameters and $z_{i,j} \sim \text{Bernoulli}(1-p)$. For regression tasks, the predictive mean and variance for a new input $\mathbf{x}^*$ can be approximated using T stochastic forward passes:

$$\mathbb{E}[y^*|\mathbf{x}^*] \approx \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}^*; \mathbf{W}_t) \tag{2.31}$$

$$\mathrm{Var}[y^*|\mathbf{x}^*] \approx \tau^{-1} + \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}^*; \mathbf{W}_t)^2 - \left(\frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}^*; \mathbf{W}_t)\right)^2 \tag{2.32}$$

where $\mathbf{W}_t$ represents the network weights with dropout applied for the t-th forward pass and τ is the model precision (inverse of the observation noise variance). This formulation decomposes the total predictive uncertainty into aleatoric uncertainty (data noise, τ^{-1}) and epistemic uncertainty (model uncertainty, captured by the variance of predictions across multiple forward passes).

In physics-informed neural networks, MC Dropout has found utility in searching for more efficient collocation points within the domain for more efficient convergence [117]. It has been used to provide UQ across spatio-temporal forecasting found within PDE modelling [118] and for getting error bounds for dynamics modelling [119].

Limitations: Despite its practical appeal and widespread adoption for uncertainty quantification in regression tasks, MC Dropout exhibits several important limitations. The uncertainty estimates provided by MC Dropout tend to be uncalibrated without additional post-processing, often underestimating true predictive uncertainty compared to full Bayesian treatments, particularly for out-of-distribution inputs where overconfidence can be problematic. The method's theoretical connection to Bayesian inference relies on restrictive assumptions about the form of approximate posterior distributions, constraining the expressiveness of the captured uncertainty. Additionally, the quality of uncertainty estimates depends critically on dropout rate and network architecture choices, with no systematic way to determine optimal values for these hyperparameters beyond heuristics or expensive cross-validation. MC Dropout also faces computational efficiency concerns during inference, as multiple forward passes (typically 50-100) are required to obtain stable uncertainty estimates, significantly increasing prediction latency in time-sensitive applications. Finally, recent research suggests that MC Dropout may fail to capture important modes of the posterior distribution, particularly in complex regression scenarios with multimodal solutions, as the implicit ensemble created by dropout tends to produce similar network behaviours rather than truly diverse function approximations that reflect the full range of plausible models given the data [120].

2.2.1.3 Deep Ensembles

Deep Ensembles represent an approximate Bayesian approach to uncertainty quantification in neural networks, leveraging the diversity of multiple independently trained networks to capture predictive uncertainty in regression tasks. This method constructs an ensemble of neural networks with identical architectures but different random initializations and trained on potentially different bootstrap samples of the data, providing a practical and scalable alternative to Bayesian methods while demonstrating competitive or superior uncertainty quantification performance, particularly for out-of-distribution detection and in capturing epistemic uncertainty arising from model misspecification or data scarcity [16].

In the Deep Ensembles framework for regression, each network in the ensemble directly outputs both the predicted mean $\mu_\theta(\mathbf{x})$ and variance $\sigma_\theta^2(\mathbf{x})$ of a Gaussian distribution for each input $\mathbf{x}$:

$$p(y|\mathbf{x},\theta) = \mathcal{N}(y|\mu_\theta(\mathbf{x}), \sigma_\theta^2(\mathbf{x})) \tag{2.33}$$

where θ represents the parameters of a neural network. The networks are trained by minimizing the negative log-likelihood (NLL) loss:

$$\mathcal{L}(\theta) = \frac{1}{2}\log\sigma_\theta^2(\mathbf{x}) + \frac{(y-\mu_\theta(\mathbf{x}))^2}{2\sigma_\theta^2(\mathbf{x})} \tag{2.34}$$

Given an ensemble of M networks with parameters $\{\theta_1, \theta_2, ..., \theta_M\}$, the predictive distribution for a new input $\mathbf{x}^*$ is approximated as a mixture of Gaussians:

$$p(y^*|\mathbf{x}^*) \approx \frac{1}{M}\sum_{m=1}^{M}\mathcal{N}(y^*|\mu_{\theta_m}(\mathbf{x}^*), \sigma_{\theta_m}^2(\mathbf{x}^*)) \tag{2.35}$$

The predictive mean and variance of this mixture distribution are given by:

$$\mathbb{E}[y^*|\mathbf{x}^*] = \frac{1}{M}\sum_{m=1}^{M}\mu_{\theta_m}(\mathbf{x}^*) \tag{2.36}$$

$$\mathrm{Var}[y^*|\mathbf{x}^*] = \frac{1}{M}\sum_{m=1}^{M}\sigma_{\theta_m}^2(\mathbf{x}^*) + \frac{1}{M}\sum_{m=1}^{M}\mu_{\theta_m}(\mathbf{x}^*)^2 - \left(\frac{1}{M}\sum_{m=1}^{M}\mu_{\theta_m}(\mathbf{x}^*)\right)^2 \tag{2.37}$$

The first term represents aleatoric uncertainty (average of predicted variances), while the second term captures epistemic uncertainty (variance of predicted means), thus providing a decomposition of the total predictive uncertainty into these two fundamental components.

Deep ensembles have found utility in providing UQ for surrogates from a range of scnarios, ranging from computational fluid dynamics [121], electromagnetics [122], weather modelling [123] and nuclear fusion [124]. The method has also proven valuable in multi-fidelity modelling scenarios, where it helps quantify uncertainty when transferring knowledge between simulation scales or between simulations and experimental data, and in active learning frameworks for neural PDEs where uncertainty estimates guide the adaptive sampling of new simulation points to efficiently explore parameter spaces in computational physics problems [125, 122].

Limitations: The computational cost of training and storing multiple independent neural networks represents a significant drawback, scaling linearly with ensemble size and becoming prohibitive for large models or resource-constrained environments, though techniques like snapshot ensembles or batch ensemble training partially mitigate this issue [126, 127]. Deep Ensembles lack the principled Bayesian foundation for uncertainty quantification, potentially missing certain types of uncertainties that would be captured by a proper posterior approximation, particularly when prior knowledge should constrain the solution space. The method's effectiveness depends critically on achieving sufficient diversity among ensemble members, which can be challenging when training converges to similar local minima despite different initialisations, limiting the ensemble's ability to explore multiple modes of the solution space in complex regression problems. Furthermore, determining the optimal ensemble size involves a trade-off between computational resources and uncertainty estimation quality, with no theoretical guidance beyond empirical validation. Finally, while Deep Ensembles often demonstrate good empirical performance for out-of-distribution detection, they can still produce overconfident predictions in certain far-from-training regions where all ensemble members make similar errors, as each network is trained to minimize the same objective function and may share similar inductive biases that do not reflect the true predictive uncertainty in these regions [128]

2.2.2 Frequentist Methods

2.2.2.1 Evidential Networks

Evidential Neural Networks (ENNs) represent a class of models that directly estimate prediction uncertainty without requiring sampling or ensemble techniques. They formulate learning as an evidence acquisition process, where training examples add support to an evidential distribution over model parameters or predictions, resulting in a single-forward-pass approach to uncertainty quantification that is computationally efficient during inference.

The core principle of evidential regression is to place evidential priors over the likelihood function parameters. For a regression task with target y, traditional neural networks predict a Gaussian likelihood with parameters $\theta = \{\mu, \sigma^2\}$. In contrast, ENNs model a Normal-Inverse-Gamma (NIG) distribution as an evidential prior over these parameters:

$$p(\mu, \sigma^2 | \gamma, \upsilon, \alpha, \beta) = \frac{\beta^\alpha \sqrt{\upsilon}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta + \upsilon(\gamma - \mu)^2}{2\sigma^2}\right\} \tag{2.38}$$

Here, a neural network outputs the evidential parameters $\{\gamma, \upsilon, \alpha, \beta\}$. From these parameters, we can directly compute: - Prediction: $\mathbb{E}[\mu] = \gamma$ - Aleatoric uncertainty: $\mathbb{E}[\sigma^2] = \frac{\beta}{\alpha-1}$ - Epistemic uncertainty: $\mathrm{Var}[\mu] = \frac{\beta}{\upsilon(\alpha-1)}$

The loss function typically includes both a negative log-likelihood term and a regularisation term $\lambda|\gamma - y_i|\Phi$ where $\Phi = 2\upsilon + \alpha$ represents the total evidence, penalising overconfidence in incorrect predictions.

Evidential regression has been successfully applied in scientific machine learning contexts, including molecular property prediction for drug discovery [129], fault diagnosis in mechanical systems citep zhou2023trustworthy, and surrogate modelling of PDEs [130]. In PDE solving, ENNs provide valuable uncertainty quantification that helps identify solution regions where predictions may be unreliable due to complex dynamics, boundary conditions, or insufficient training data, allowing researchers to make informed decisions about model trustworthiness without the computational overhead of sampling-based approaches.

Limitations: Despite their advantages, evidential neural networks face limitations including sensitivity to the regularisation coefficient λ, potential underestimation of uncertainty in high-dimensional spaces, and degraded calibration in transfer learning scenarios. Additionally, the theoretical foundations are less established than fully Bayesian approaches. ENNs may still produce overconfident predictions in some adversarial settings or under significant distribution shifts, requiring careful validation against traditional UQ methods for safety-critical applications.

2.2.2.2 Conformal Prediction

Conformal Prediction (CP) is a distribution-free uncertainty quantification framework that constructs prediction intervals with rigorous coverage guarantees. Unlike Bayesian or ensemble methods that often rely on distributional assumptions, CP provides valid statistical guarantees for finite samples without requiring model or distribution assumptions.

Given i.i.d. samples $(X_i, Y_i) \sim P$, $i = 1, \ldots, n$, where $X_i \in \mathcal{X}$ are inputs and $Y_i \in \mathbb{R}$ are real-valued outputs, the goal is to construct a prediction interval $\hat{C}_n(x)$ for a new input x such that:

$$P(Y_{n+1} \in \hat{C}_n(X_{n+1}) | (X_i, Y_i)_{i=1}^n) \geq 1 - \alpha \tag{2.39}$$

where (X_{n+1}, Y_{n+1}) is a new data point and $\alpha \in (0, 1)$ is a user-specified error rate. In regression settings, the split conformal algorithm operates as follows: (1) divide data into training and calibration sets; (2) train a regression model $\hat{f}$ on the training set; (3) define a nonconformity score function, typically $s(x,y) = |y - \hat{f}(x)|$; (4) compute scores on calibration data: $s_i = s(X_i, Y_i)$; (5) calculate the $(1-\alpha)$-quantile $\hat{q} = \text{Quantile}(s_1, ..., s_n; \lceil (n+1)(1-\alpha) \rceil / n)$; and (6) for a new input x, construct the prediction interval $\hat{C}(x) = [\hat{f}(x) - \hat{q}, \hat{f}(x) + \hat{q}]$.

In scientific machine learning and PDE solving, CP has demonstrated considerable utility for regression tasks requiring reliable uncertainty estimates. CP has found utility in providing UQ for dynamical systems [131], PINNs [132], DeepONets [133] as well as functional calibration for operater learning [134].

Limitations CP provides only marginal coverage guarantees rather than conditional coverage, meaning the prediction intervals may be unnecessarily wide in some regions of the input space and too narrow in others. The method requires a separate calibration dataset which reduces available training data, and the quality of the prediction intervals heavily depends on the underlying regression model's accuracy—poor models lead to uniformly wide intervals with limited practical utility. For time series or sequential regression problems with distribution drift, standard CP may fail to maintain coverage unless properly modified to account for temporal dependencies using techniques like weighted conformal prediction, which introduces additional complexity and hyperparameters that require careful tuning.

2.2.2.3 Bootstrapping Ensembles

Bootstrapping ensembles represent a practical non-parametric approach to uncertainty quantification (UQ) in neural network regression by training multiple models on resampled datasets, leveraging the statistical principle of bootstrapping to estimate predictive uncertainty without requiring modifications to the underlying network architecture or regression loss function, making it particularly attractive for regression tasks where quantifying prediction confidence intervals is essential.

Given a regression dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^d$ are input features and $y_i \in \mathbb{R}$ are continuous target values, the bootstrap ensemble approach operates as follows:

1. Create M bootstrap datasets $\{\mathcal{D}_m\}_{m=1}^{M}$, where each $\mathcal{D}_m$ is generated by sampling N points from $\mathcal{D}$ with replacement
2. Train M identical regression models $\{f_{\theta_m}(\mathbf{x})\}_{m=1}^{M}$, each on a different bootstrap dataset $\mathcal{D}_m$
3. For a test point $\mathbf{x}_*$, compute:

   Mean: $\mu(\mathbf{x}_*) = \frac{1}{M}\sum_{m=1}^{M} f_{\theta_m}(\mathbf{x}_*)$

   Variance: $\sigma^2(\mathbf{x}_*) = \frac{1}{M}\sum_{m=1}^{M} (f_{\theta_m}(\mathbf{x}_*) - \mu(\mathbf{x}_*))^2$

where $\mu(\mathbf{x}_*)$ represents the ensemble's predicted value at $\mathbf{x}_*$, and $\sigma^2(\mathbf{x}_*)$ quantifies the epistemic uncertainty (model uncertainty) in the regression prediction, enabling the construction of confidence intervals as $\mu(\mathbf{x}_*) \pm z_{\alpha/2} \cdot \sigma(\mathbf{x}_*)$, with $z_{\alpha/2}$ being the critical value for the desired confidence level.

In scientific regression contexts, bootstrapping ensembles have demonstrated significant utility across multiple domains including: modelling epidemic breakouts [135], predicting outputs of photovoltaic cells [136], and providing uncertainty for financial derivatives modelling [137].

Limitations: The computational cost scales linearly with ensemble size, requiring training and storing multiple regression models; they primarily capture epistemic uncertainty from data sampling variations but may inadequately represent aleatoric uncertainty inherent in noisy regression data; small or imbalanced datasets may lead to unreliable uncertainty estimates due to insufficient sampling diversity; individual ensemble members may converge to different local minima in complex regression landscapes, potentially overestimating predictive uncertainty; ensemble variance often requires post-hoc calibration to provide statistically valid confidence intervals for regression predictions, particularly in extrapolation regions; and the method assumes independent and identically distributed data points, which may not hold for spatiotemporal regression problems with complex correlation structures.

2.2.3 Verified Methods

2.2.3.1 Interval Arithmetic

Interval arithmetic (IA) provides rigorous uncertainty bounds for neural networks by replacing standard floating-point operations with interval operations that track lower and upper bounds. For regression tasks, IA propagates intervals through each layer, ensuring that true function values always remain contained within the computed intervals, thereby offering guaranteed error bounds on predictions without requiring statistical assumptions [108].

The core principle involves representing values as intervals $[a,b]$ where $a \leq b$, and defining operations that maintain correctness:

$$[a,b]+[c,d]=[a+c,b+d] \tag{2.40}$$

$$[a,b]-[c,d]=[a-d,b-c] \tag{2.41}$$

$$[a,b]\times[c,d]=[\min(ac,ad,bc,bd),\max(ac,ad,bc,bd)] \tag{2.42}$$

$$[a,b]\ /\ [c,d]=[a,b]\times[1/d,1/c] \text{ when } 0\notin[c,d] \tag{2.43}$$

For neural networks with ReLU activations, Interval Bound Propagation (IBP) extends these principles to matrix operations:

$$\mathbf{z}^{(l)}=\mathbf{W}^{(l)}\mathbf{a}^{(l-1)}+\mathbf{b}^{(l)} \tag{2.44}$$

$$\mathbf{z}_{\text{lower}}^{(l)}=\mathbf{W}_{+}^{(l)}\mathbf{a}_{\text{lower}}^{(l-1)}+\mathbf{W}_{-}^{(l)}\mathbf{a}_{\text{upper}}^{(l-1)}+\mathbf{b}^{(l)} \tag{2.45}$$

$$\mathbf{z}_{\text{upper}}^{(l)}=\mathbf{W}_{+}^{(l)}\mathbf{a}_{\text{upper}}^{(l-1)}+\mathbf{W}_{-}^{(l)}\mathbf{a}_{\text{lower}}^{(l-1)}+\mathbf{b}^{(l)} \tag{2.46}$$

$$\mathbf{a}^{(l)}=\text{ReLU}(\mathbf{z}^{(l)}) \tag{2.47}$$

$$\mathbf{a}_{\text{lower}}^{(l)}=\text{ReLU}(\mathbf{z}_{\text{lower}}^{(l)}) \tag{2.48}$$

$$\mathbf{a}_{\text{upper}}^{(l)}=\text{ReLU}(\mathbf{z}_{\text{upper}}^{(l)}) \tag{2.49}$$

where $\mathbf{W}_{+}^{(l)}$ and $\mathbf{W}_{-}^{(l)}$ denote the positive and negative components of the weight matrix respectively.

In regression contexts, IA has been applied to provide verified bounds on neural network approximations of complex functions, partial differential equations, and dynamical systems [138, 139]. They have found utility in propagating input uncertainty across safety-critical systems [140] and short-term windspeed prediction [141].

Limitations: The primary limitation of interval arithmetic for neural networks is the dependency problem, where multiple occurrences of the same variable are treated as independent, leading to overestimation of bounds that grows exponentially with network depth and width. For regression tasks, this manifests as excessively conservative uncertainty intervals that may be too wide for practical use. Fundamental theoretical limits, established in [142], demonstrate that certain simple specifications cannot be proven using interval analysis alone, regardless of network architecture. These limitations have motivated advanced methods that maintain correlations between variables, such as affine arithmetic and zonotopes, though these come with increased computational complexity.

2.2.3.2 Taylor Models

Taylor models provide a rigorous approach to quantifying uncertainty in neural networks by representing functions as a multivariate polynomial plus a bounded remainder interval. This approach combines interval arithmetic with symbolic computation to track functional dependencies accurately, making it particularly powerful for regression tasks where guaranteed bounds on neural network outputs are required [109].

A d-dimensional Taylor model of order k is defined as a tuple $T = (p, \Delta, D)$ where $p = (p_1, \dots, p_d)^T$ is a vector of multivariate polynomials $p_i : D \to \mathbb{R}$ of degree at most k, $\Delta = \Delta_1 \times \cdots \times \Delta_d$ is a hyperrectangle containing the origin that bounds the remainder, and $D \subseteq \mathbb{R}^d$ is the domain. For a neural network function f, a Taylor model ensures that $f(x) \in p_n(x) + \Delta$ for all $x \in D$, providing guaranteed bounds on the network's predictions that account for approximation errors.

In regression contexts, Taylor models enable rigorous uncertainty quantification in neural networks by providing guaranteed error bounds rather than probabilistic estimates. They have been successfully applied to validate neural surrogate models for physical systems, verify neural network controllers in safety-critical applications, and certify the behaviour of physics-informed neural networks solving partial differential equations. Recent advances in [143] allow for automatically computing sharp bounds on the Taylor remainder series, making this approach increasingly practical for complex neural architectures.

Limitations The primary challenges of Taylor models for neural network uncertainty quantification include: computational complexity that scales exponentially with input dimension, limiting applicability to high-dimensional problems; the wrapping effect where dependencies between variables are lost during operations, leading to overestimation when propagating through deep networks; difficulty in handling non-polynomial activation functions which require additional approximation steps; and significant computational overhead compared to non-verified approaches such as Bayesian neural networks or dropout. Recent work integrating Taylor models with zonotopes [144] has addressed some of these limitations by preserving dependencies across control cycles in neural network verification tasks.

2.2.3.3 Zonotopes

Zonotope-based methods provide formal guarantees for neural network uncertainty quantification through geometric set representations that can efficiently capture non-convex output distributions. These methods compute tight enclosures of the regression function outputs by abstracting input-output relations using polynomial approximations, making them particularly suitable for safety-critical applications where guaranteed bounds on predictions are essential [145].

A zonotope $Z \subset \mathbb{R}^n$ is defined as:

$$Z = \left\{ c + \sum_{i=1}^{p} \xi_i g_i \mid \xi_i \in [-1, 1] \right\} \tag{2.50}$$

where $c \in \mathbb{R}^n$ is the centre vector and $g_i \in \mathbb{R}^n$ are the generator vectors that define the geometry of the set. For neural network regression, polynomial zonotopes extend this representation to capture more complex non-linear behaviours:

$$PZ = \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E(k,i)} \right) G_{(:,i)} + \sum_{j=1}^{q} \beta_j G_{I(:,j)} \mid \alpha_k, \beta_j \in [-1, 1] \right\} \tag{2.51}$$

Here, $G \in \mathbb{R}^{n \times h}$ represents dependent generators, $G_I \in \mathbb{R}^{n \times q}$ represents independent generators, and $E \in \mathbb{N}_0^{p \times h}$ is an exponent matrix. This formulation allows polynomial zonotopes to be closed under polynomial maps, enabling precise propagation through neural networks with various activation functions [146].

Polynomial zonotopes have demonstrated significant advantages in regression tasks for scientific computing, particularly for providing guaranteed confidence bands around predictions in critical domains [147]. They excel in quantifying uncertainty for surrogate models like Fourier Neural Operators, solving differential equations such as Burger's equation, where both aleatoric and epistemic uncertainties need guaranteed bounds [148]. They have also found utility in verifying Lagrangian neural networks [149]. The method allows efficient propagation of input uncertainties through complex neural network architectures while preserving dependencies between inputs and outputs, which is crucial for multi-dimensional regression problems where correlation structures should be maintained.

2.2.4 Limitations

The primary limitations of zonotope-based uncertainty quantification methods involve computational scalability with increasing network depth. The representation size of polynomial zonotopes grows substantially when propagated through deep networks, necessitating periodic order reduction that can introduce additional over-approximation. Computing tight bounds on regression outputs becomes computationally expensive for high-dimensional problems, requiring approximation techniques that balance precision and efficiency. Recent progress has shown that this can be circumvented by using dimensionality reduction techniques such as singular value decomposition [148]. While the approach excels at capturing non-convex behaviours in regression outputs, it still requires considerable expertise to tune parameters control-ling the trade-off between computational efficiency and tightness of the uncertainty bounds, particularly when applying the method to complex scientific regression tasks with highly non-linear behaviours.

Bibliography

[1] OpenFOAM Foundation. Openfoam: The open source cfd toolbox, 2025. Accessed: 2025-04-15.

[2] Guillaume Giudicelli, Alexander Lindsay, Logan Harbour, Casey Icenhour, Mengnan Li, Joshua E. Hansel, Peter German, Patrick Behne, Oana Marin, Roy H. Stogner, Jason M. Miller, Daniel Schwen, Yaqi Wang, Lynn Munday, Sebastian Schunert, Benjamin W. Spencer, Dewen Yushu, Antonio Recuero, Zachary M. Prince, Max Nezdyur, Tianchen Hu, Yinbin Miao, Yeon Sang Jung, Christopher Matthews, April Novak, Brandon Langley, Timothy Truster, Nuno Nobre, Brian Alger, David Andrš, Fande Kong, Robert Carlsen, Andrew E. Slaughter, John W. Peterson, Derek Gaston, and Cody Permann. 3.0 - MOOSE: Enabling massively parallel multiphysics simulations. *SoftwareX*, 26:101690, 2024.

[3] M. Hoelzl, G. T. A. Huijsmans, S. J. P. Pamela, M. Bécoulet, E. Nardon, F.J. Artola, B. Nkonga, C. V. Atanasiu, V. Bandaru, A. Bhole, D. Bonfiglio, A. Cathey, O. Czarny, A. Dvornova, T. Fehér, A. Fil, E. Franck, S. Futatani, M. Gruca, H. Guillard, J. W. Haverkort, I. Holod, D. Hu, S. K. Kim, S. Q. Korving, L. Kos, I. Krebs, L. Kripner, G. Latu, F. Liu, P. Merkel, D. Meshcheriakov, V. Mitterauer, S. Mochalskyy, J. A. Morales, R. Nies, N. Nikulsin, F. Orain, J. Pratt, R. Ramasamy, P. Ramet, C. Reux, K. Särkimäki, N. Schwarz, P. Singh Verma, S. F. Smith, C. Sommariva, E. Strumberger, D. C. van Vugt, M. Verbeek, E. Westerhof, F. Wieschollek, and J. Zielinski. The jorek non-linear extended mhd code and applications to large-scale instabilities and their con-

trol in magnetically confined fusion plasmas. *Nuclear Fusion*, 61(6):065001, 2021.

[4] G. Danabasoglu, J.-F. Lamarque, J. Bacmeister, D. A. Bailey, A. K. DuVivier, J. Edwards, L. K. Emmons, J. Fasullo, R. Garcia, A. Gettelman, C. Hannay, M. M. Holland, W. G. Large, P. H. Lauritzen, D. M. Lawrence, J. T. M. Lenaerts, K. Lindsay, W. H. Lipscomb, M. J. Mills, R. Neale, K. W. Oleson, B. Otto-Bliesner, A. S. Phillips, W. Sacks, S. Tilmes, L. van Kampenhout, M. Vertenstein, A. Bertini, J. Dennis, C. Deser, C. Fischer, B. Fox-Kemper, J. E. Kay, D. Kinnison, P. J. Kushner, V. E. Larson, M. C. Long, S. Mickelson, J. K. Moore, E. Nienhouse, L. Polvani, P. J. Rasch, and W. G. Strand. The community earth system model version 2 (cesm2). *Journal of Advances in Modeling Earth Systems*, 12(2):e2019MS001916, 2020. e2019MS001916 2019MS001916.

[5] J. A. K. Horwitz. Estimating the carbon footprint of computational fluid dynamics. *Physics of Fluids*, 36(4):045109, 04 2024.

[6] G. D. Smith. *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford University Press, Oxford, 3 edition, 1985.

[7] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Addison-Wesley, Reading, MA, 1995.

[8] J. N. Reddy. *Introduction to the Finite Element Method*. McGraw-Hill Education, New York, 3 edition, 2006.

[9] Bengt Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, Cambridge, UK, 1996.

[10] Alexander Lavin, David Krakauer, Hector Zenil, Justin Gottschlich, Tim Mattson, Johann Brehmer, Anima Anandkumar, Sanjay Choudry, Kamil Rocki, Atılım Güneş Baydin, Carina Prunkl, Brooks Paige, Olexandr Isayev, Erik Peterson, Peter L. McMahon, Jakob Macke, Kyle Cranmer, Jiaxin Zhang,

Haruko Wainwright, Adi Hanuka, Manuela Veloso, Samuel Assefa, Stephan Zheng, and Avi Pfeffer. Simulation intelligence: Towards a new generation of scientific methods. *arXiv:2112.03235*, 2021.

[11] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[12] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

[13] Vignesh Gopakumar, Stanislas Pamela, Lorenzo Zanisi, Zongyi Li, Ander Gray, Daniel Brennand, Nitesh Bhatia, Gregory Stathopoulos, Matt Kusner, Marc Peter Deisenroth, Anima Anandkumar, the JOREK Team, and MAST Team. Plasma surrogate modelling using fourier neural operators. *Nuclear Fusion*, 64(5):056025, 4 2024.

[14] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

[15] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

[16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017.

[17] Vignesh Gopakumar, Ander Gray, Joel Oskarsson, Lorenzo Zanisi, Stanislas Pamela, Daniel Giles, Matt Kusner, and Marc Peter Deisenroth. Uncertainty quantification of surrogate models using conformal prediction, 2024.

[18] Cristian Bodnar, Wessel Bruinsma, Ana Lucic, Megan Stanley, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan Weyn, Haiyu Dong, Anna Vaughan, Jayesh Gupta, Kit Tambiratnam, Alex Archibald, Elizabeth Heider, Max Welling, Richard Turner, and Paris Perdikaris. Aurora: A foundation model of the atmosphere. Technical Report MSR-TR-2024-16, Microsoft Research AI for Science, May 2024.

[19] Apostolos F. Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023.

[20] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[21] 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.

[22] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling, 2019.

[23] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020.

[24] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, Jul 2022.

[25] 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, Dec 2021.

[26] Byoungchan Jang, Alan A. Kaptanoglu, Rahul Gaur, Shaowu Pan, Matt Landreman, and William Dorland. Grad–shafranov equilibria via data-free physics informed neural networks. *Physics of Plasmas*, 31(3):032510, Mar 2024.

[27] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, December 2018.

[28] Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, Mar 2018.

[29] Ameya D. Jagtap, Ehsan Kharazmi, and George Em 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.

[30] Vignesh Gopakumar, Stanislas Pamela, and Debasmita Samaddar. Loss landscape engineering via data regulation on pinns. *Machine Learning with Applications*, 12:100464, 2023.

[31] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective, 2020.

[32] Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training pinns: A loss landscape perspective, 2024.

[33] Francisco Sahli Costabal, Simone Pezzuto, and Paris Perdikaris. δ-pinns: Physics-informed neural networks on complex geometries. *Engineering Applications of Artificial Intelligence*, 127:107324, January 2024.

[34] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

[35] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

[36] A. Mathews, M. Francisquez, J. W. Hughes, D. R. Hatch, B. Zhu, and B. N. Rogers. Uncovering turbulent plasma dynamics via deep learning from partial observations. *Phys. Rev. E*, 104:025205, Aug 2021.

[37] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

[38] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks, 2020.

[39] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks, 2019.

[40] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

[41] Taco S. Cohen and Max Welling. Group equivariant convolutional networks, 2016.

[42] Pierre-Yves Lagrave and Eliot Tron. Equivariant neural networks and differential invariants theory for solving partial differential equations. *Physical Sciences Forum*, 5(1), 2022.

[43] David M Knigge, David Wessels, Riccardo Valperga, Samuele Papa, Jan-Jakob Sonke, Erik J Bekkers, and Stratis Gavves. Space-time continuous PDE forecasting using equivariant neural fields. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[44] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data, 2018.

[45] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018.

[46] Marius Kurz, Andrea Beck, and Benjamin Sanderse. Harnessing equivariance: Modeling turbulence with graph neural networks, 2025.

[47] Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical cnns, 2018.

[48] Jonathan A. Weyn, Dale R. Durran, and Rich Caruana. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, 12(9):e2020MS002109, 2020. e2020MS002109 10.1029/2020MS002109.

[49] Oliver T. Unke, Mihail Bogojeski, Michael Gastegger, Mario Geiger, Tess Smidt, and Klaus-Robert Müller. Se(3)-equivariant prediction of molecular wavefunctions and electronic densities, 2021.

[50] Johannes Brandstetter, Max Welling, and Daniel E. Worrall. Lie point symmetry data augmentation for neural pde solvers, 2022.

[51] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks, 2022.

[52] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

[53] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3165–3176. PMLR, 13–18 Jul 2020.

[54] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[55] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations, 2020.

[56] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations, 2020.

[57] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, Mar 2021.

[58] Bogdan Raonić, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes, 2023.

[59] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Lno: Laplace neural operator for solving differential equations, 2023.

[60] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems.

Computer Methods in Applied Mechanics and Engineering, 404:115783, 2023.

[61] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[62] Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC ’23, New York, NY, USA, 2023. Association for Computing Machinery.

[63] Louis Serrano, Lise Le Boudec, Armand Kassaï Koupaï, Thomas X Wang, Yuan Yin, Jean-Noël Vittaut, and patrick gallinari. Operator learning with neural fields: Tackling PDEs on general geometries. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[64] Sanchit Bedi, Karn Tiwari, Prathosh A. P., Sri Harsha Kota, and N. M. Anoop Krishnan. A neural operator for forecasting carbon monoxide evolution in cities. *npj Clean Air*, 1(1):2, Mar 2025.

[65] Gege Wen, Zongyi Li, Qirui Long, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M. Benson. Real-time high-resolution CO_2 geological storage prediction using nested fourier neural operators. *Energy & Environmental Science*, 16(4):1732–1741, 2023.

[66] Khemraj Shukla, Vivek Oommen, Ahmad Peyvan, Michael Penwarden, Nicholas Plewacki, Luis Bravo, Anindya Ghoshal, Robert M. Kirby, and George Em Karniadakis. Deep neural operators as accurate surrogates

for shape optimization. *Engineering Applications of Artificial Intelligence*, 129:107615, 2024.

[67] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM / IMS J. Data Sci.*, 1(3), may 2024.

[68] Vignesh Gopakumar, Stanislas Pamela, Lorenzo Zanisi, Zongyi Li, Anima Anandkumar, and MAST Team. Fourier neural operator for plasma modelling, 2023.

[69] N. Carey, L. Zanisi, S. Pamela, V. Gopakumar, J. Omotani, J. Buchanan, J. Brandstetter, F. Paischer, G. Galletti, and P. Setinek. Data efficiency and long-term prediction capabilities for neural operator surrogate models of edge plasma simulations, 2025.

[70] S. J. P. Pamela, N. Carey, J. Brandstetter, R. Akers, L. Zanisi, J. Buchanan, V. Gopakumar, M. Hoelzl, G. Huijsmans, K. Pentland, T. James, G. Antonucci, and the JOREK Team. Neural-parareal: Dynamically training neural operators as coarse solvers for time-parallelisation of fusion mhd simulations, 2024.

[71] Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes, 2024.

[72] Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining codomain attention neural operators for solving multiphysics pdes. *Advances in Neural Information Processing Systems*, 37, 2024.

[73] Benedikt Alkin, Andreas Fürst, Simon Lucas Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A

framework for efficiently scaling neural operators. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[74] Samuel Lanthaler, Andrew M. Stuart, and Margaret Trautner. Discretization error of fourier neural operators, 2024.

[75] Francesca Bartolucci, Emmanuel de Bezenac, Bogdan Raonic, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[76] Jean Kossaifi, Nikola Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized fourier neural operator for high-resolution pdes, 2023.

[77] Yolanne Yi Ran Lee. Autoregressive renaissance in neural pde solvers, 2023.

[78] Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. Apebench: A benchmark for autoregressive neural emulators of pdes, 2024.

[79] Anthony Zhou and Amir Barati Farimani. Predicting change, not states: An alternate framework for neural pde surrogates, 2024.

[80] Shawn G Rosofsky, Hani Al Majed, and E A Huerta. Applications of physics informed neural operators. *Machine Learning: Science and Technology*, 4(2):025022, may 2023.

[81] Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-informed deep neural operator networks, 2022.

[82] Colin White, Julius Berner, Jean Kossaifi, Mogab Elleithy, David Pitt, Daniel Leibovici, Zongyi Li, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operators with exact differentiation on arbitrary geometries. In *The Symbiosis of Deep Learning and Differential Equations III*, 2023.

[83] 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*, 36(4), April 2024.

[84] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[85] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1994.

[86] Eric Rowland Lalonde, Benjamin Vischschraper, Girma Bitsuamlak, and Kaoshan Dai. Comparison of neural network types and architectures for generating a surrogate aerodynamic wind turbine blade model. *Journal of Wind Engineering and Industrial Aerodynamics*, 216:104696, 2021.

[87] Pierre Baldi, Kyle Cranmer, Taylor Faucett, Peter Sadowski, and Daniel Whiteson. Parameterized neural networks for high-energy physics. *The European Physical Journal C*, 76(5):235, 04 2016.

[88] Petr Mánek, Graham Van Goffrier, Vignesh Gopakumar, Nikos Nikolaou, Jonathan Shimwell, and Ingo Waldmann. Fast regression of the tritium breeding ratio in fusion reactors. *Machine Learning: Science and Technology*, 4(1):015008, 2023.

[89] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, 1986.

[90] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[91] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[92] Vignesh Gopakumar, Stanislas Pamela, and Lorenzo Zanisi. Fourier-rnns for modelling noisy physics data, 2023.

[93] Yihao Hu, Tong Zhao, Shixin Xu, Zhiliang Xu, and Lizhen Lin. Neural-pde: A rnn based neural network for solving time dependent pdes, 2022.

[94] Priyabrata Saha, Saurabh Dash, and Saibal Mukhopadhyay. Physics-incorporated convolutional recurrent neural networks for source identification and forecasting of dynamical systems, 2021.

[95] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[96] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[97] Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized PDE modeling. *Transactions on Machine Learning Research*, 2023.

[98] Vignesh Gopakumar and Debasmita Samaddar. Image mapping the temporal evolution of edge characteristics in tokamaks using neural networks. *Machine Learning: Science and Technology*, 1(1):015006, 2020.

[99] Quang Tuyen Le and Chinchun Ooi. Surrogate modeling of fluid dynamics with a multigrid inspired neural network architecture. *Machine Learning with Applications*, 6:100176, 2021.

[100] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[101] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023.

[102] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for physical surrogate models. In *NeurIPS 2023 AI for Science Workshop*, 2023.

[103] Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K. Gupta, and Aditya Grover. Climax: A foundation model for weather and climate, 2023.

[104] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 05 1992.

[105] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression, 2020.

[106] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

[107] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[108] T. Hickey, Q. Ju, and M. H. Van Emden. Interval arithmetic: From principles to implementation. *J. ACM*, 48(5):1038–1068, September 2001.

[109] Kyoko Makino and Martin Berz. Taylor models and other validated functional inclusion methods. In *2003 International Journal of Pure and Applied Mathematics*, 2003.

[110] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2018.

[111] Kevin Linka, Amelie Schäfer, Xuhui Meng, Zongren Zou, George Em Karniadakis, and Ellen Kuhl. Bayesian physics informed neural networks for real-world nonlinear dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 402:115346, 2022. A Special Issue in Honor of the Lifetime Achievements of J. Tinsley Oden.

[112] Zongren Zou, Xuhui Meng, Apostolos F. Psaros, and George E. Karniadakis. Neuraluq: A comprehensive library for uncertainty quantification in neural differential equations and operators. *SIAM Review*, 66(1):161–190, 2024.

[113] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, January 2021.

[114] Dylan Sam, Rattana Pukdee, Daniel P. Jeong, Yewon Byun, and J. Zico Kolter. Bayesian neural networks with domain knowledge priors, 2024.

[115] Christophe Bonneville and Christopher Earls. Bayesian deep learning for partial differential equation parameter discovery with sparse and noisy data. *Journal of Computational Physics: X*, 16:100115, 2022.

[116] Xuhui Meng, Hessam Babaee, and George Em Karniadakis. Multi-fidelity bayesian neural networks: Algorithms and applications. *Journal of Computational Physics*, 438:110361, 2021.

[117] Yuri Aikawa, Naonori Ueda, and Toshiyuki Tanaka. Improving the efficiency of training physics-informed neural networks using active learning. *New Generation Computing*, 42(4):739–760, Nov 2024.

[118] Abouzar Choubineh, Jie Chen, Frans Coenen, and Fei Ma. Applying monte carlo dropout to quantify the uncertainty of skip connection-based convolutional neural networks optimized by big data. *Electronics*, 12(6), 2023.

[119] Xinyue Xu and Julian Wang. Comparative analysis of physics-guided bayesian neural networks for uncertainty quantification in dynamic systems. *Forecasting*, 7(1), 2025.

[120] Loic Le Folgoc, Vasileios Baltatzis, Sujal Desai, Anand Devaraj, Sam Ellis, Octavio E. Martinez Manzanera, Arjun Nair, Huaqi Qiu, Julia Schnabel, and Ben Glocker. Is mc dropout bayesian?, 2021.

[121] Rakesh Halder, Mohammadmehdi Ataei, Hesam Salehipour, Krzysztof Fidkowski, and Kevin Maki. Reduced-order modeling of unsteady fluid flow using neural network ensembles. *Physics of Fluids*, 36(7), July 2024.

[122] Raphaël Pestourie, Youssef Mroueh, Chris Rackauckas, Payel Das, and Steven G. Johnson. Physics-enhanced deep surrogates for partial differential equations. *Nature Machine Intelligence*, 5(12):1458–1465, December 2023.

[123] Sebastian Scher and Gabriele Messori. Ensemble methods for neural network-based weather forecasts, 2021.

[124] L. Zanisi, A. Ho, J. Barr, T. Madula, J. Citrin, S. Pamela, J. Buchanan, F.J. Casson, V. Gopakumar, and JET Contributors. Efficient training sets for surrogate models of tokamak turbulence with active deep ensembles. *Nuclear Fusion*, 64(3):036022, feb 2024.

[125] Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active learning for neural pde solvers, 2024.

[126] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: An alternative approach to efficient ensemble and lifelong learning, 2020.

[127] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free, 2017.

[128] Vignesh Gopakumar, Ander Gray, Lorenzo Zanisi, Timothy Nunn, Stanislas Pamela, Daniel Giles, Matt J. Kusner, and Marc Peter Deisenroth. Calibrated physics-informed uncertainty quantification, 2025.

[129] Ava P Soleimany, Alexander Amini, Samuel Goldman, Daniela Rus, Sangeeta N Bhatia, and Connor W Coley. Evidential deep learning for guided molecular property prediction and discovery. *ACS central science*, 7(8):1356–1367, 2021.

[130] Hai Siong Tan, Kuancheng Wang, and Rafe McBeth. Evidential physics-informed neural networks, 2025.

[131] Aoming Liang, Qi Liu, Lei Xu, Fahad Sohrab, Weicheng Cui, Changhui Song, and Moncef Gabbouj. Conformal prediction on quantifying uncertainty of dynamic systems, 2024.

[132] Lena Podina, Mahdi Torabi Rad, and Mohammad Kohandel. Conformalized physics-informed neural networks, 2024.

[133] Christian Moya, Amirhossein Mollaali, Zecheng Zhang, Lu Lu, and Guang Lin. Conformalized-deeponet: A distribution-free framework for uncertainty quantification in deep operator networks. *Physica D: Nonlinear Phenomena*, 471:134418, 2025.

[134] Ziqi Ma, Kamyar Azizzadenesheli, and Anima Anandkumar. Calibrated uncertainty quantification for operator learning via conformal prediction. *arXiv preprint arXiv:2402.01960*, 2024.

[135] Gerardo Chowell and Ruiyan Luo. Ensemble bootstrap methodology for forecasting dynamic growth processes using differential equations: application

to epidemic outbreaks. *BMC Medical Research Methodology*, 21(1):34, Feb 2021.

[136] Zahi M. Omer and Hussain Shareef. Adaptive boosting and bootstrapped aggregation based ensemble machine learning methods for photovoltaic systems output current prediction. In *2019 29th Australasian Universities Power Engineering Conference (AUPEC)*, pages 1–6, 2019.

[137] Ryno du Plooy and Pierre J. Venter. A comparison of artificial neural networks and bootstrap aggregating ensembles in a modern financial derivative pricing framework. *Journal of Risk and Financial Management*, 14(6), 2021.

[138] Krasymyr Tretiak, Georg Schollmeyer, and Scott Ferson. Neural network model for imprecise regression with interval dependent variables. *Neural Networks*, 161:550–564, 2023.

[139] D Chetwynd, K Worden, and G Manson. An application of interval-valued neural networks to a regression problem. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 462(2074):3097–3114, 2006.

[140] David Betancourt and Rafi L. Muhanna. Interval deep learning for computational mechanics problems under input uncertainty. *Probabilistic Engineering Mechanics*, 70:103370, 2022.

[141] Ronay Ak, Valeria Vitelli, and Enrico Zio. An interval-valued neural network approach for uncertainty quantification in short-term wind speed prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 26(11):2787–2800, 2015.

[142] Matthew Mirman, Maximilian Baader, and Martin Vechev. The fundamental limits of interval arithmetic for neural networks, 2021.

[143] Matthew Streeter and Joshua V. Dillon. Automatically bounding the taylor remainder series: Tighter bounds and new applications, 2023.

[144] Christian Schilling, Marcelo Forets, and Sebastián Guadalupe. Verification of neural-network control systems by integrating taylor models and zonotopes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):8169–8177, June 2022.

[145] Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. *Open- and Closed-Loop Neural Network Verification Using Polynomial Zonotopes*, page 16–36. Springer Nature Switzerland, 2023.

[146] Tobias Ladner and Matthias Althoff. Exponent relaxation of polynomial zonotopes and its applications in formal neural network verification. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024.

[147] Augustin Lemesle, Julien Lehmann, and Tristan Le Gall. Neural network verification with pyrat, 2024.

[148] Ander Gray, Vignesh Gopakumar, Sylvain Rousseau, and Sébastien Destercke. Guaranteed confidence-band enclosures for pde surrogates, 2025.

[149] Matt Jordan, Jonathan Hayase, Alexandros G. Dimakis, and Sewoong Oh. Zonotope domains for lagrangian neural network verification, 2022.