ASME Journal of Computing and Information Science in Engineering
Online journal at:
https://asmedigitalcollection.asme.org/computingengineering

Salah A. Faroughi[1]
Geo-Intelligence Laboratory,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666
e-mail: salah.faroughi@txstate.edu

Nikhil M. Pawar
Geo-Intelligence Laboratory,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666

Célio Fernandes
Geo-Intelligence Laboratory,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666;
Centre of Mathematics (CMAT),
University of Minho,
Campus of Gualtar,
4710-057 Braga, Portugal

Maziar Raissi
Department of Applied Mathematics,
University of Colorado Boulder,
Boulder, CO 610101

Subasish Das
Artificial Intelligence in Transportation Lab,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666

Nima K. Kalantari
Computer Science and Engineering Department,
Texas A&M University,
College Station, TX 77843

Seyed Kourosh Mahjour
Geo-Intelligence Laboratory,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666

Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks and Operators in Scientific Computing: Fluid and Solid Mechanics

Advancements in computing power have recently made it possible to utilize machine learning and deep learning to push scientific computing forward in a range of disciplines, such as fluid mechanics, solid mechanics, materials science, etc. The incorporation of neural networks is particularly crucial in this hybridization process. Due to their intrinsic architecture, conventional neural networks cannot be successfully trained and scoped when data are sparse, which is the case in many scientific and engineering domains. Nonetheless, neural networks provide a solid foundation to respect physics-driven or knowledge-based constraints during training. Generally speaking, there are three distinct neural network frameworks to enforce the underlying physics: (i) physics-guided neural networks (PgNNs), (ii) physics-informed neural networks (PiNNs), and (iii) physics-encoded neural networks (PeNNs). These methods provide distinct advantages for accelerating the numerical modeling of complex multiscale multiphysics phenomena. In addition, the recent developments in neural operators (NOs) add another dimension to these new simulation paradigms, especially when the real-time prediction of complex multiphysics systems is required. All these models also come with their own unique drawbacks and limitations that call for further fundamental research. This study aims to present a review of the four neural network frameworks (i.e., PgNNs, PiNNs, PeNNs, and NOs) used in scientific computing research. The state-of-the-art architectures and their applications are reviewed, limitations are discussed, and future research opportunities are presented in terms of improving algorithms, considering causalities, expanding applications, and coupling scientific and deep learning solvers. [DOI: 10.1115/1.4064449]

Keywords: physics-guided neural networks, physics-informed neural networks, physics-encoded neural networks, solid mechanics, fluid mechanics, machine learning, deep learning, scientific computing, artificial intelligence, data-driven engineering, machine learning for engineering applications, multiphysics modeling and simulation, physics-based simulations

1 Introduction

Machine learning (ML) and deep learning (DL) are emerging as pivotal technologies to propel scientific research and computing across a range of fields, including but not limited to fluid mechanics [1], solid mechanics [2], materials science [3,4], and so on. The advent of multi-teraflop machines equipped with thousands of processors designed for scientific computing, in conjunction with advanced sensory-based experimentation, has led to an exponential growth of structured and unstructured heterogeneous data in science and engineering disciplines. ML and DL methodologies were initially introduced to address the challenge of developing efficient data modeling procedures, which impeded scientists' ability to interact with heterogeneous and complex data swiftly [5]. These approaches have demonstrated the potential to transform scientific computing by enabling the exploration of vast design spaces, identifying multidimensional connections, and resolving ill-posed issues [6–8].

However, conventional ML and DL methods are unable to extract interpretative information and expertise from complex

[1]Corresponding author.

Manuscript received April 27, 2023; final manuscript received December 4, 2023; published online January 29, 2024. Assoc. Editor: Guang Lin.

multidimensional data. They may be effective in mapping observational or computational data, but their predictions may be physically irrational or dubious, resulting in poor generalization [9–11]. For this reason, scientists initially considered these methodologies as a magic black box devoid of a solid mathematical foundation and incapable of interpretation. Notwithstanding, learning techniques constitute a new paradigm for accurately solving scientific and practical problems orders of magnitude faster than conventional solvers.

Deep learning (i.e., neural networks (NNs) mimicking the human brain) and scientific computing share common historical and intellectual links that are normally unrealized, e.g., differentiability [9]. Figure 1 shows a schematic representation of the history of development for a plethora of scientific computing and DL approaches (only seminal works are included). In the last decade, breakthroughs in DL and computing power have enabled the use of DL in a broad variety of scientific computing, especially in fluid mechanics [1,11–13], solid mechanics [2,14–16], and materials science [17–20], albeit at the cost of accuracy and loss of generality [21]. These data-driven methods are routinely applied to fulfill one of the following goals: (i) accelerate direct numerical simulations using surrogate modeling [22], (ii) accelerate adjoint sensitivity analysis [9] and uncertainty quantification [23], (iii) accelerate probabilistic programming [24], and (iv) accelerate inverse problems [25,26]. For example, in the first goal, the physical parameters of the system (e.g., dimensions, mass, momentum, temperature) are used as inputs to predict the next state of the system or its effects (i.e., outputs), and in the last goal, the outputs of a system (e.g., a material with targeted properties) are used as inputs to infer the intrinsic physical attributes that meet the requirements (i.e., the model's outputs). To accomplish these goals, lightweight DL models can be constructed to partially or fully replace a bottleneck step in the scientific computing processes [21,27,28].

Due to the intrinsic architecture of conventional DL methods, their learning is limited to the scope of the datasets with which the training is conducted (e.g., specific boundary conditions, material types, spatiotemporal discretization), and inference cannot be successfully scoped under any unseen conditions (e.g., new geometries, new material types, new boundary conditions). Because the majority of the scientific fields are not (big) data-oriented domains and cannot provide comprehensive datasets that cover all possible conditions, these models trained based on sparse datasets are accelerated but not predictive [28]. Thus, it is logical to leverage the wealth of prior knowledge, underlying physics, and domain expertise to further constrain these models while training on available and sparse data points. NNs are better suited to digest physical-driven or knowledge-based constraints during training. Based on how underlying physics is incorporated, the authors classified neural network applications in scientific computing into three separate types: (i) physics-guided neural networks (PgNNs), (ii) physics-informed neural networks (PiNNs), and (iii) physics-encoded neural networks (PeNNs).

In PgNN-based models, standard supervised DL techniques are used to construct surrogate mappings between formatted inputs and outputs that are generated using experiments and computations in a controlled setting and curated through extensive processes to ensure compliance with physics principles and fundamental rules [28,47]. Such models require a large and sufficient dataset to be trained and used reliably. A PgNN-based model maps a set of inputs $\mathbf{x}$ to a related set of outputs $\mathbf{y}$ using an appropriate function $\mathbf{F}$ with unknown parameters $\mathbf{w}$ such that $\mathbf{y} = F(\mathbf{x}; \mathbf{w})$. By specifying a particular structure for F, a data-driven approach generally attempts to fine-tune the parameters $\mathbf{w}$ so that the overall error between true values, $\hat{\mathbf{y}}$, and those from model predictions, $\mathbf{y}$, is minimized [8]. For complex physical systems, the data are likely sparse due to the high cost of data acquisition [48]. The majority of state-of-the-art PgNNs lack robustness and fail to fulfill any guarantees of generalization (i.e., interpolation [44,49] and extrapolation [50]). To remedy this issue, PiNNs have been introduced to perform supervised learning tasks while obeying given laws of physics in the form of general nonlinear differential equations [7,11,51–53].

The PiNN-based models respect the physical laws by incorporating a weakly imposed loss function consisting of the residuals of physics equations and boundary constraints. To differentiate the neural network outputs with respect to their inputs (i.e., spatiotemporal coordinates and model parameters), these methods leverage automatic differentiation (AD) [54]. By minimizing the loss function, the neural network can closely approximate the solution

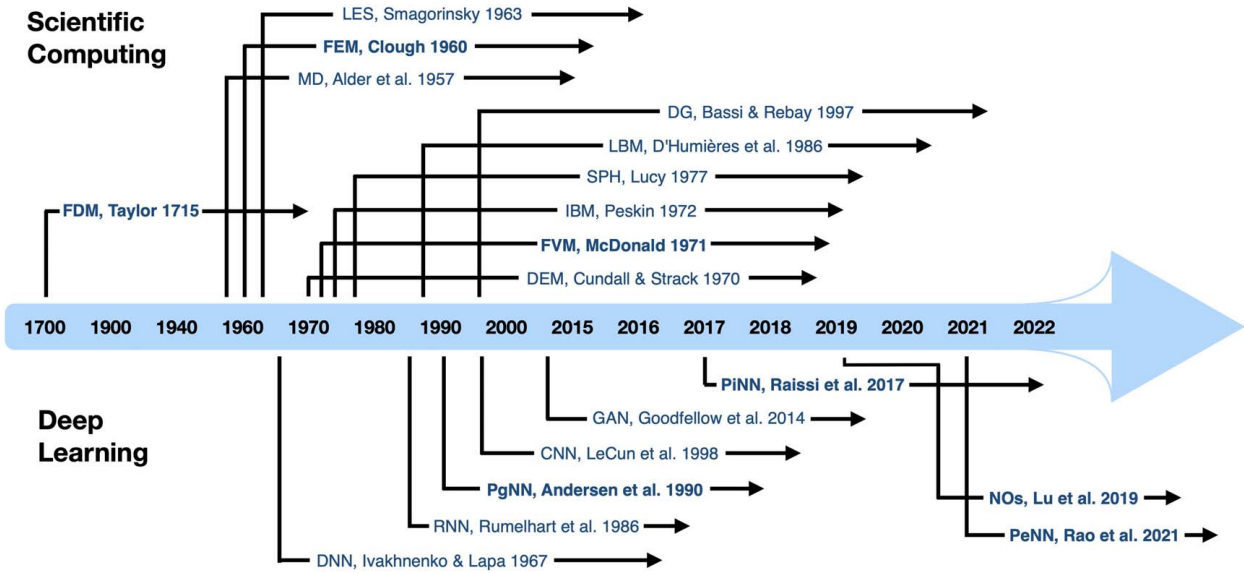

Fig. 1 A schematic representation of the history of development, including only seminal works, for scientific computing and DL approaches. For scientific computing, the following are listed: finite difference method (FDM) [29], molecular dynamics (MD) [30], finite element method (FEM) [31], large Eddy simulation (LES) [32], discrete element method (DEM) [33], finite volume method (FVM) [34], immersed boundary method (IBM) [35], smoothed particle hydrodynamics (SPH) [36], lattice Boltzmann method (LBM) [37], and discontinuous Galerkin (DG) [38]. For deep learning, the following are listed: deep neural network (DNN) [39], recurrent neural network (RNN) [40], physics-guided neural network (PgNN) [41], convolutional neural network (CNN) [42], generative adversarial network (GAN) [43], physics-informed neural network (PiNN) [44], neural operators (NOs) [45], and physics-encoded neural network (PeNN) [46].

[55,56]. As a result, PiNNs lay the groundwork for a new modeling and computation paradigm that enriches DL with long-standing achievements in mathematical physics [44,51]. The PiNN models face a number of limitations relating to theoretical considerations (e.g., convergence and stability [7,57,58]) and implementation considerations (e.g., neural network design, boundary condition management, and optimization aspects) [11,46]. In addition, in cases where the explicit form of differential equations governing the complex dynamics is not fully known a priori, PiNNs encounter serious limitations [59]. For such cases, another family of DL approaches known as PeNNs has been proposed [46].

The PeNN-based models leverage advanced architectures to address issues with data sparsity and the lack of generalization encountered by both PgNNs and PiNNs models. PeNN-based models forcibly encode known physics into their core architecture (e.g., NeuralODE [60]). By construction, PeNN-based models extend the learning capability of a neural network from instance learning (imposed by PgNN and PiNN architectures) to continuous learning [60]. The encoding mechanisms of the underlying physics in PeNNs are fundamentally different from those in PiNNs [61,62], although they can be integrated to achieve the desired nonlinearity of the model. Compared to PgNNs and PiNNs, the neural networks generated by the PeNN paradigm offer better performance against data sparsity and model generalizability [46].

There is another family of supervised learning methods that do not fit well in the PgNN, PiNN, and PeNN categories, as defined earlier. These models, dubbed neural operators (NOs), learn the underlying linear and nonlinear continuous operators, such as integrals and fractional Laplacians, using advanced architectures (e.g., deep operator networks (DeepONets) [45,63]). The data-intensive learning procedure of a neural operator may resemble the PgNN-based model learning, as both enforce the physics of the problem using labeled input–output dataset pairs. However, a neural operator is very different from a PgNN-based model that lacks generalization properties due to under-parameterization. A neural operator can be combined with the PiNN and PeNN methods to train a model that can learn complex nonlinearity in physical systems with extremely high generalization accuracy [50]. The robustness of this combination (e.g., PiNN and neural operators [64,65]) for applications requiring real-time inference is a distinguishing characteristic. This review article is primarily intended for the scientific computing community interested in the application of neural networks in computational fluid and solid mechanics. It discusses the general architectures, advantages, and limitations of PgNNs, PiNNs, PeNNs, and neural operators and reviews the most prominent applications of these methods in fluid and solid mechanics. The remainder of this work is structured as follows: In Sec. 2, the potential of PgNNs to accelerate scientific computing is discussed. Section 3 provides an overview of PiNNs and discusses their potential to advance PgNNs. In Sec. 4, several leading PeNN architectures are discussed to address critical limitations in PgNN and PiNNs. Section 5 reviews recent developments in neural operators. Finally, in Sec. 6, an outlook for future research directions is provided.

2 Physics-Guided Neural Networks

PgNNs use standard supervised DL models to statistically learn the known physics of a desired phenomenon by extracting features or attributes from training datasets obtained through well-controlled experiments from complex dynamical systems and well-calibrated computations from multiscale multiphysics systems [66]. PgNNs consist of one or a combination of multilayer perceptron (MLP; alternatively called artificial neural networks (ANNs) or deep neural networks (DNNs) in different studies relevant to this review) [66], convolutional neural network (CNN) [66], RNN [66], GAN [67], and graph neural networks (GRNNs) [68]. Although GAN models are classified as unsupervised learning, they can be classified as PgNNs, in the context of this review article, because their underlying training is framed as a supervised learning problem [67,69]. A schematic representation of a sample PgNN architecture is illustrated in Fig. 2. Any physical problem includes a set of independent features or input features as $\mathbf{x} = [X_1, X_2, X_3, \ldots, X_n]$ and a set of dependent variables or desired outputs as $\mathbf{y} = [Y_1, Y_2, Y_3, \ldots, Y_n]$. Data describing this physical phenomenon can be generated by experimentation (e.g., sensor-based observation, etc.), closure laws (e.g., Fourier's law, Darcy's law, drag force), or the solution of governing ordinary differential equations (ODEs) and/or partial differential equations (PDEs), e.g., Burger's equation, Navier–Stokes equations, and so on. The dependent

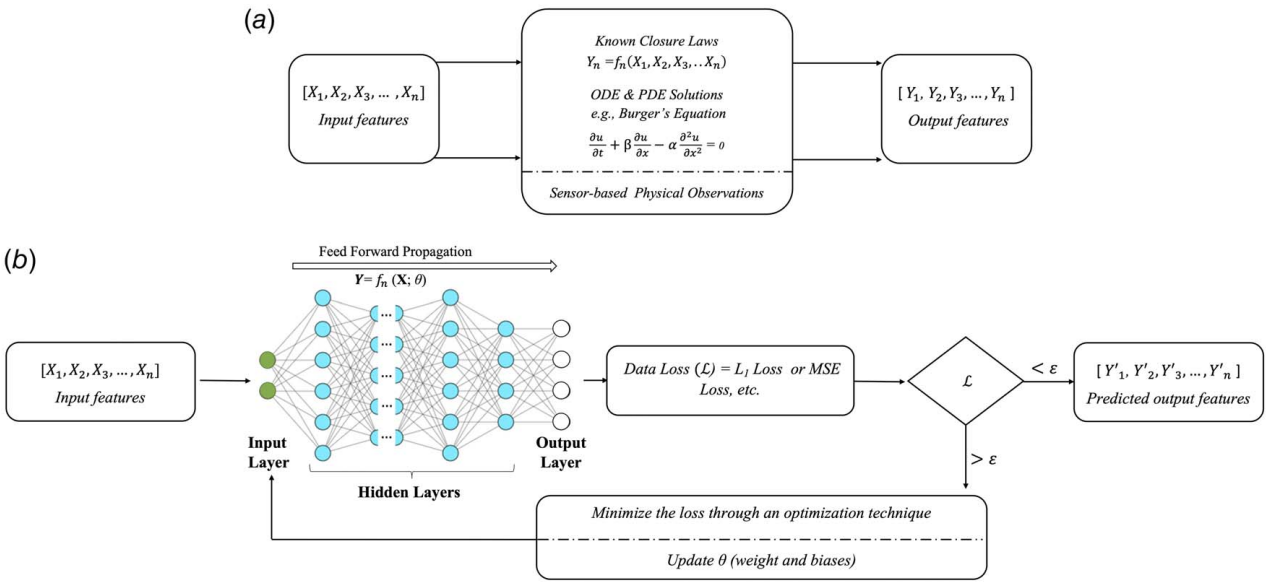

Fig. 2 A schematic architecture of PgNNs. (*a*) The typical generation of training datasets using known closure laws, direct numerical simulation of PDEs and ODEs, or experimentation to comply with physical principles. (*b*) The architecture of a PgNN model consisting of a simple feed-forward neural network (which can be replaced with any other network type). The loss function made of L_1 and L_2 regularization, MSE, or other user-defined error functions is minimized iteratively in the training phase. θ is the learnable parameter corresponding to weights/biases in the neural network that can be learned simultaneously while minimizing the loss function.

variables and independent features thus comply with physics principles, and the trained neural network is guided inherently by physics throughout training.

In PgNNs, the neurons of each layer are linked to those of the subsequent layer through a set of weights. An activation function (such as ReLU, Tanh, Sigmoid, and Linear) is then applied to the weighted sum of the outputs of the neurons in the previous layer, along with an extra bias, to obtain the output of each node [70]. This procedure sequentially obtains the output of the neurons in each layer, starting with the input. This process is typically called forward propagation. A loss function (or, alternatively, a cost function) is subsequently defined and calculated in order to evaluate the accuracy of the prediction. Commonly used loss functions for regression are L1 [71] and mean-squared error (MSE) [71]. The next step in training involves error backpropagation, which calculates the partial derivatives/gradients of the cost function with respect to weights and biases (i.e., θ as shown in Fig. 2). Finally, an optimization technique, such as gradient descent, stochastic gradient descent, adaptive moment estimation (Adam), or mini-batch gradient descent [72,73], is used to minimize the loss function and simultaneously compute and update θ parameters using the calculated gradients from the backpropagation procedure. The process is iterated until the desired level of accuracy is obtained for a PgNN.

In recent years, PgNN has been widely adopted to accelerate computational fluid dynamics (CFD) [74], computational solid mechanics [75], and multifunctional material designs [76]. PgNN has been utilized in all computationally intensive and time-consuming aspects of scientific computing, including (i) preprocessing [74,77,78], such as mesh generation; (ii) discretization and modeling [79–81], such as finite difference method (FDM), finite volume method (FVM), finite element method (FEM), discrete element method, and molecular dynamics (MD); and (iii) postprocessing, such as output assimilation and visualization [82–84]. These studies are conducted to (i) train shallow networks on small datasets to replace a bottleneck (i.e., a computationally expensive step) in conventional forward numerical modeling, for example, the calculation of the drag coefficient in complex fluid flow modeling [28,85–88]; or (ii) train relatively deep networks on larger datasets generated for a particular problem, such as targeted sequence design within the coarse-grained polymer genome [89]. These networks take into account the physical principles underlying the training data and accelerate the simulation process [28,84].

Although PgNNs training appears to be straightforward, generating the data by tackling the underlying physics for complex physical problems could require substantial computational cost [7,15]. Once trained, a PgNN can significantly accelerate the computation speed for the phenomena of interest. It is worth noting that, while a PgNN model may achieve good accuracy on the training set based on numerous attempts, it is more likely to memorize the trends, noise, and detail in the training set rather than intuitively comprehend the pattern in the dataset. This is one of the reasons why PgNNs lose their prediction ability when inferred/tested outside the scope of the training datasets. The overfitting of PgNN can be mitigated in different ways [90–92] to enhance the predictability of the model within the scope of the training data. In the following subsections, we review the existing literature and highlight some of the most recent studies that applied PgNNs to accelerate different steps in scientific computing for applications in fluid and solid mechanics.

2.1 Preprocessing. Preprocessing is often the most work-intensive component in scientific computing, regardless of the numerical model type (e.g., FEM, FDM, FVM). The main steps in this component are the disassembly of the domain into small, but finite, parts (i.e., mesh generation, evaluation, and optimization), and the upscaling and/or downscaling of the mesh properties to use a spatiotemporally coarse mesh while implicitly solving for unresolved fine-scale physics. These two steps are time consuming and require expert-level knowledge; hence, they are potential candidates to be replaced by accelerated PgNN-based models.

2.1.1 Mesh Generation. Mesh generation is a critical step for numerical simulations. Zhang et al. [77] proposed the automatic generation of an unstructured mesh based on the prediction of the required local mesh density throughout the domain. For this purpose, an ANN was trained to guide a standard mesh generation algorithm. They also proposed extending the study to other architectures, such as CNN or GRNN, for future studies including larger datasets and/or higher-dimensional problems. Huang et al. [74] adopted a DL approach to identify optimal mesh densities. They generated optimized meshes using classical CFD tools (e.g., Simcenter STAR-CCM+ [93]) and proposed training a CNN to predict optimal mesh densities for arbitrary geometries. The addition of an adaptive mesh refinement version accelerated the overall process without compromising accuracy and resolution. The authors proposed learning optimal meshes (generated by the corresponding solvers with adjoint functionality) using ANN, which may be used as a starting point in other simulation tools irrespective of the specific numerical approach [74]. Wu et al. [78] also proposed a mesh optimization method by integrating the moving mesh method with DL in order to solve the mesh optimization problem. With the experiments carried out, a neural network with high accuracy was constructed to optimize the mesh while preserving the specified number of nodes and topology of the initial given mesh. By using this technique, they also demonstrated that the moving mesh algorithm is independent of the CFD computation [78].

In mesh generation, a critical issue has been the evaluation of mesh quality due to a lack of general and effective criteria. Chen et al. [94] presented a benchmark dataset (i.e., the NACA-Market reference dataset) to facilitate the evaluation of the quality of a mesh. They presented GridNet, a technique that uses a deep CNN to perform an automatic evaluation of the mesh quality. This method receives the mesh as input and performs the evaluation. The evaluation of mesh quality using a deep CNN model trained on the NACA-Market dataset proved to be viable with an accuracy of 92.5% [94].

2.1.2 Cross-Scaling Techniques. It is always desirable to numerically solve a multiphysics problem on a spatiotemporally coarser mesh to minimize computational cost. For this reason, different upscaling [95,96], downscaling [97], and cross-scaling [98] methods have been developed to determine accurate numerical solutions to nonlinear problems across a broad range of length- and time-scales. One viable choice is to use a coarse mesh that reliably depicts long-wavelength dynamics and accounts for unresolved small-scale physics. Deriving the mathematical model (e.g., boundary conditions) for coarse representations, on the other hand, is relatively difficult. Bar-Sinai et al. [96] proposed a PgNN model to learn optimal PDE approximations based on actual solutions to known underlying equations. The ANN outputs spatial derivatives which are then optimized to best satisfy the equations on a low-resolution grid. Compared to typical discretization methods (e.g., finite difference), the recommended ANN method was considerably more accurate while integrating the set of nonlinear equations at a resolution that was four to eight times coarser [96]. The main challenge in this approach, however, is to systematically derive these kinds of solution-adaptive discrete operators. Maddu et al. [95] developed a PgNN, dubbed STENCIL-NET, for learning resolution-specific local discretization of nonlinear PDEs. By combining spatially and temporally adaptive parametric pooling on regular Cartesian grids with knowledge about discrete-time integration, STENCIL-NET can accomplish numerically stable discretization of the operators for any arbitrary nonlinear PDE. The STENCIL-NET model can also be used to determine PDE solutions over a spatiotemporal scale wider than the training dataset. In their article, the authors employed STENCIL-NET for long-term forecasting of chaotic PDE solutions on coarse spatiotemporal grids to test their hypothesis. Comparing the STENCIL-NET model with baseline numerical techniques (e.g., fully vectorized WENO [99]), the predictions on

Table 1 A nonexhaustive list of recent studies that leveraged PgNNs to accelerate the preprocessing part in scientific computing

Area of application	NN type	Objective	Reference
Mesh generation	ANN	Generating unstructured mesh	[77]
	CNN	Predicting meshes with optimal density and accelerating meshing process without compromising performance or resolution	[74]
	ANN	Generating high-quality tetrahedral meshes	[100]
	ANN	Developing a mesh generator tool to produce high-quality FEM meshes	[101]
	ANN	Generating finite element mesh with less complexities	[102]
Mesh evaluation	CNN	Conducting automatic mesh evaluation and quality assessment	[94]
Mesh optimization	ANN	Optimizing mesh while retaining the same number of nodes and topology as the initially given mesh	[78]
	ANN	Enhancing mesh density for stress fields in plane-strain problems.	[103]
Cross-scaling	ANN	Utilizing data-driven discretization to estimate spatial derivatives that are tuned to best fulfill the equations on a low-resolution grid.	[96]
	ANN (STENCIL-NET)	Providing solution-adaptive discrete operators to predict PDE solutions on bigger spatial domains and for longer time frames than it was trained	[95]

coarser grids were faster by up to 25–150 times on GPUs and 2–14 times on CPUs, while maintaining the same accuracy [95].

Table 1 reports a nonexhaustive list of recent works that leveraged PgNNs to accelerate the preprocessing part of scientific computing. These studies collectively concluded that PgNN can be successfully integrated to achieve a considerable speedup factor in mesh generation, mesh evaluation, and cross-scaling, which are vital for many complex problems explored using scientific computing techniques. Section 2.2 discusses the potential of PgNN to be incorporated into the modeling components, hence yielding a higher speedup factor or greater accuracy.

2.2 Modeling and Postprocessing

2.2.1 PgNNs for Fluid Mechanics. PgNN has gained considerable attention from the fluid mechanics community. The study by Lee and Chen [104] on estimating fluid properties using ANN was among the first studies to apply PgNN to fluid mechanics. Since then, the application of PgNNs in fluid mechanics has been extended to a wide range of applications, e.g., laminar and turbulent flows, non-Newtonian fluid flows, aerodynamics, especially to speed up the traditional computational fluid dynamics (CFD) solvers.

For incompressible laminar flow simulations, the numerical procedure to solve the Navier–Stokes equations is considered the main bottleneck. To alleviate this issue, PgNNs have been used as part of the resolution process. For example, Yang et al. [105] proposed a novel data-driven projection method using an ANN to avoid iterative computation of the projection step in grid-based fluid simulations. The efficiency of the proposed data-driven projection method was shown to be significant, especially in large-scale fluid flow simulations. Tompson et al. [106] used a CNN to predict numerical solutions to the inviscid Euler equations for fluid flows. Unsupervised training that incorporates multiframe information was proposed to improve long-term stability. The CNN model produced very stable divergence-free velocity fields with improved accuracy compared to those obtained by the commonly used Jacobi method [107]. Chen et al. [108] later developed a U-net-based architecture, a particular case of a CNN model, for the prediction of velocity and pressure field maps around arbitrary 2D shapes in laminar flows. The CNN model is trained with a dataset composed of random shapes constructed using Bézier curves and then by solving the Navier–Stokes equations using a CFD solver. The predictive efficiency of the CNN model was also assessed on unseen shapes, using ad hoc error functions, specifically, the MSE levels for these predictions were found to be of the same order of magnitude as those obtained on the test subset, i.e., between 1.0×10^{-5} and 5.0×10^{-5} for both pressure and velocity, respectively.

Moving from laminar to turbulent flow regimes, PgNNs have also been widely used for the formulation of turbulence closure models [109]. For example, Ling et al. [110] utilized a feed-forward MLP and a specialized neural network to predict Reynolds-averaged Navier–Stokes (RANS) and large Eddy simulation (LES) turbulence problems. Their specialized neural network embeds Galilean invariance [111] using a higher-order multiplicative layer. The performance of this model was compared with that of MLP and ground truth simulations. They concluded that the specialized neural network can predict the anisotropy tensor on an invariant tensor basis, resulting in significantly more accurate predictions than MLP. Maulik et al. [112] presented a closure framework for subgrid modeling of Kraichnan turbulence [113]. To determine the dynamic closure strength, the proposed framework used an implicit map with inputs such as grid-resolved variables and eddy viscosities. Training an ANN with extremely subsampled data obtained from high-fidelity direct numerical simulations (DNSs) yields the optimal map. The ANN model was found to be successful in imbuing the decaying turbulence problem with dynamic kinetic energy dissipation, allowing accurate capture of coherent structures and inertial range fidelity. Later, Kim and Lee [114] used simple linear regression, SLinear, multiple linear regression, MLinear, and a CNN to predict turbulent heat transfer (i.e., the wall-normal heat flux, q_w) using other wall information, including the streamwise wall-shear stress, spanwise wall-shear stress or streamwise vorticity, and pressure fluctuations, obtained by DNSs of a channel flow (see Fig. 3(*a*)). The constructed network was trained using adaptive moment estimation (ADAM) [115,116], and the grid searching method [117,118] was performed to optimize the depth and the width of the CNN. Their finding showed that the PgNN model is less sensitive to the input resolution, indicating its potential as a good heat flux model in turbulent flow simulation. Yousif et al. [119] also proposed an efficient method for generating turbulent inflow conditions based on a PgNN formed by a combination of a multiscale convolutional autoencoder with a subpixel convolution layer (MSCSP-AE) [120,121] and long short-term memory (LSTM) [122,123] model. The proposed model was found to have the capability to deal with spatial mapping of turbulent flow fields.

PgNNs have also been applied in the field of aerodynamics. Kou and Zhang [124] presented a review paper on typical data-driven methods, including system identification, feature extraction, and data fusion, that have been used to model unsteady aerodynamics. The efficacy of these data-driven methods is described by several benchmark cases in aeroelasticity. Wang et al. [125] described the application of ANN to the modeling of the swirling flow field in a combustor (see Fig. 3(*b*)). Swirling flow field data from particle image velocimetry (PIV) was used to train an ANN model. The trained PgNN model was successfully tested to predict the swirling

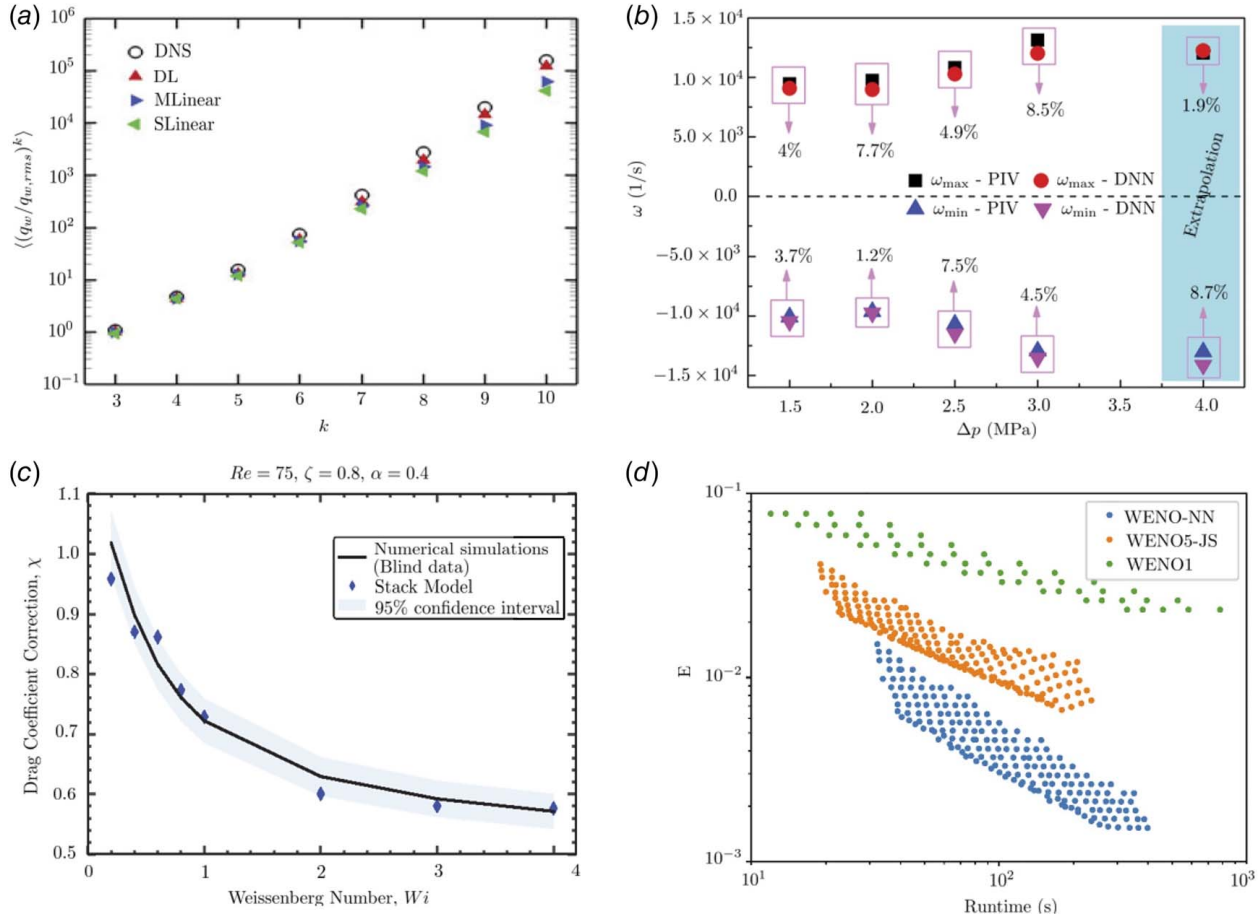

Fig. 3 **(*a*) A comparison of the high-order moments of the heat flux data obtained from DNS of turbulent heat transfer and predictions obtained by SLinear, MLinear, and CNN (i.e., PgNN) models developed by Kim and Lee [114]. Notice that $q_{w,rms}$ is the root-mean-square error (RMSE) of q_w, k denotes the index of the weights in the network, and the angle bracket denotes the average over all test points. (*b*) A comparison of a PgNN model and the PIV technique for predicting the swirling flow field in a combustor [125]. The changes in maximum and minimum vorticity, ω (1/s), in a swirling flow field are shown for several pressure drops, Δp (MPa). (c) The performance of the PgNN model developed by Faroughi et al. [28] against the blind dataset generated to predict the drag coefficient of a spherical particle translating in a Giesekus fluid at Reynolds number Re = 75, retardation ratio $\zeta = 0.8$, and mobility parameter $\alpha = 0.4$. (*d*) A comparison of the L_2^2 error (*E*) and simulation run time of WENO-NN, weighted ENO-Jiang Shu (WENO5-JS) scheme convergent in the fifth order, and weighted ENO (WENO1) scheme convergent in the first order, to simulate shock wave interactions [131].**

flow field under unknown inlet conditions. Chowdhary et al. [126] studied the efficacy of combining ANN models with projection-based model reduction techniques [127,128] to develop an ANN-surrogate model for computationally expensive high-fidelity physics models, specifically for complex hypersonic turbulent flows. The surrogate model was used to perform a Bayesian estimation of the freestream conditions and parameters of the shear stress transport turbulence model. The surrogate model was then embedded in the high-fidelity (Reynolds-averaged Navier–Stokes) flow simulator, using shock-tunnel data. Siddiqui et al. [129] developed a nonlinear data-driven model, encompassing time delay neural networks, for a pitching wing. The pitch angle was considered as input to the model, while the lift coefficient was considered as output. The results showed that the trained models were able to capture the nonlinear aerodynamic forces more accurately than linear and semi-empirical models, especially at higher offset angles. Wang et al. [130] also proposed a multifidelity reduced-order model based on multitask learning ANNs to efficiently predict the unsteady aerodynamic performance of an iced airfoil. The results indicated that the proposed model achieves higher accuracy and better generalization capability compared to single-fidelity and single-task modeling approaches.

The simulation of complex fluid flows, specifically using fluids that exhibit viscoelastic nature and nonlinear rheological behaviors, is another topic where PgNNs have been applied [132–134]. The dynamics of these fluids is generally governed by nonlinear constitutive equations that lead to stiff numerical problems [135,136]. Faroughi et al. [28] developed a PgNN model to predict the drag coefficient of a spherical particle translating in viscoelastic fluids (see Fig. 3(*c*)). The PgNN considered a stacking technique (i.e., ensembling random forrest [137], extreme gradient boosting [138], and ANN models) to digest inputs (Reynolds number, Weissenberg number, viscosity ratio, and mobility factor considering both Oldroyd-B and Giesekus fluids), and outputs drag predictions based on individual learner's predictions and an ANN meta-regressor. The accuracy of the model was successfully checked against blind datasets generated by DNSs. Lennon et al. [139] also developed a tensor basis neural network (TBNN) allowing rheologists to construct learnable constitutive models that incorporate essential physical information while remaining agnostic to details regarding particular experimental protocols or flow kinematics. The TBNN model incorporates a universal approximator within a materially objective tensorial constitutive framework that, by construction, respects physical constraints, such as frame-invariance and tensor symmetry, required by continuum mechanics. Due to the embedded TBNN, the developed rheological universal differential equation quickly learns simple yet accurate and highly general

models to describe the provided training data, allowing for rapid discoery of constitutive equations.

Lastly, PgNNs have also been extensively used to improve both the accuracy and speed of CFD solvers. Stevens and Colonius [131] developed a DL model (the weighted essentially nonoscillatory neural network (WENO-NN)) to enhance a finite volume method used to discretize PDEs with discontinuous solutions such as the turbulence–shock wave interactions (see Fig. 3(*d*)). Kochkov et al. [22] used hybrid discretizations, combining CNNs and subcomponents of a numerical solver, to interpolate differential operators onto a coarse mesh with high accuracy. The training of the model was performed within a standard numerical method to solve the underlying PDEs as a differentiable program, and the method allows for end-to-end gradient-based optimization of the entire algorithm. The method learns accurate local operators for convective fluxes and residual terms and matches the accuracy of an advanced numerical solver running at 8–10 times finer resolution while performing the computation 40–80 times faster. Cai et al. [140] implemented a least-squares ReLU neural network (LSNN) to solve the linear advection–reaction problem with a discontinuous solution. They showed that the proposed method outperformed the mesh-based numerical methods in terms of the number of degrees-of-freedom. Haber et al. [141] suggested an autoencoder CNN to reduce the resolution cost of a scalar transport equation coupled to the Navier–Stokes equations. Lara and Ferrer [142] proposed to accelerate high-order discontinuous Galerkin methods using neural networks. The methodology and bounds were examined for a variety of meshes, polynomial orders, and viscosity values for the 1D Burgers' equation. List et al. [143] employed CNN to train turbulence models to improve under-resolved, low-resolution solutions to the incompressible Navier–Stokes equations at simulation time. The developed method consistently outperforms simulations with a twofold higher resolution in both spatial and temporal dimensions. For mixing layer cases, the hybrid model on average resembles the performance of threefold reference simulations, which corresponds to a speedup of 7.0 times for the temporal layer and 3.7 times for the spatial mixing layer.

Table 2 reports a nonexhaustive list of recent studies that leveraged PgNN to model fluid flow problems. These studies collectively concluded that PgNNs can be successfully integrated with CFD solvers or used as standalone surrogate models to develop accurate and yet faster modeling components for scientific computing in fluid mechanics. In the next section, the potential application of PgNNs in computational solid mechanics is discussed.

2.2.2 PgNNs for Solid Mechanics. PgNNs have also been extensively adopted by the computational solid mechanics community [147]. The study by Andersen et al. [41] on welding modeling using ANN was among the first studies that applied PgNN to solid mechanics. Since then, the application of PgNN has been extended to a wide range of problems, e.g., structural analysis, topology optimization, inverse materials design and modeling, health condition assessment, especially to speed up the traditional forward and inverse modeling methods in computational mechanics.

In the area of structural analysis, Tadesse et al. [148] proposed an ANN to predict the mid-span deflections of a composite bridge with flexible shear connectors. The ANN was tested on six different bridges, yielding a maximum root-mean-square error (RMSE) of 3.79%, which can be negligible in practice. In addition, the authors developed closed-form solutions based on ANN that can be used to quickly predict deflection in everyday design. In a study by Güneyisi et al. [149], ANN was employed to develop a new formulation for the flexural overstrength factor for steel beams, using 141 experimental data samples with different cross-sectional typologies for model training. The model achieved a comparable training and testing accuracy of 99%, indicating its reliability in estimating beams' over-strength. Hung et al. [150] used ANN to predict the ultimate load factor of a nonlinear, inelastic steel truss, using the cross sections of the members as input and load factor as output. A planar 39-bar steel truss was used to demonstrate the efficiency of the proposed ANN. The ANN-based model yielded a high degree of accuracy, with an average loss of less than 0.02, in predicting the ultimate load factor of the nonlinear inelastic steel truss. Chen et al. [151] also used ANN to solve a three-dimensional (3D) inverse problem of a collision between an elastoplastic hemispherical metal shell and a rigid impactor. The goal was to predict the position, velocity, and duration of the collision based on the permanent plastic deformation of the shell. For static and dynamic loading, the ANN model predicted location, velocity, and collision duration with high accuracy. Hosseinpour et al. [152] used PgNN to assess the buckling capacity of castellated steel beams subjected to lateral-distortional buckling. As shown in Fig. 4(*a*), the ANN-based

Table 2 A nonexhaustive list of studies that leveraged PgNNs to model fluid computational flow problems

Area of application	NN type	Objective	Reference
Laminar flows	CNN	Calculating numerical solutions to the inviscid Euler equations	[106]
	CNN	Predicting the velocity and pressure fields around arbitrary 2D shapes	[108]
Turbulent flows	ANN	Developing a model for the Reynolds stress anisotropy tensor using high-fidelity simulation data	[110]
	CNN	Designing and training artificial neural networks based on local convolution filters for LES	[144]
	ANN	Developing subgrid modelling of Kraichnan turbulence	[112]
	CNN	Estimating turbulent heat transfer based on other wall information acquired from channel flow DNSs	[114]
	CNN-LSTM	Generating turbulent inflow conditions with accurate statistics and spectra	[119]
Aerodynamics	CNN-MLP	Predicting incompressible laminar steady flow field over airfoils	[145]
	ANN	Modeling the swirling flow field in a combustor	[125]
	PCA-ANN	Creating surrogate models of computationally expensive, high-fidelity physics models for complex hypersonic turbulent flows	[126]
	ANN	Predicting unsteady aerodynamic performance of iced airfoil	[130]
Viscoelastic flows	ANN	Predicting drag coefficient of a spherical particle translating in viscoelastic fluids	[28]
	ANN	Constructing learnable constitutive models using a universal approximator within a materially objective tensorial constitutive framework	[139]
Enhance CFD solvers	ANN	Developing an improved finite volume method for simulating PDEs with discontinuous solutions	[131]
	CNN	Interpolating differential operators onto a coarse mesh with high accuracy	[22]
	LSNN	Solving the linear advection–reaction problem with discontinuous solution	[140]
	CNN	Accelerating high-order discontinuous Galerkin methods	[142]
	CNN	Developing turbulence model to improve under-resolved low-resolution solutions to the incompressible Navier–Stokes equations at simulation time	[143]
	ANN	Estimating the fluid field for a steady-state hydraulic valve	[146]

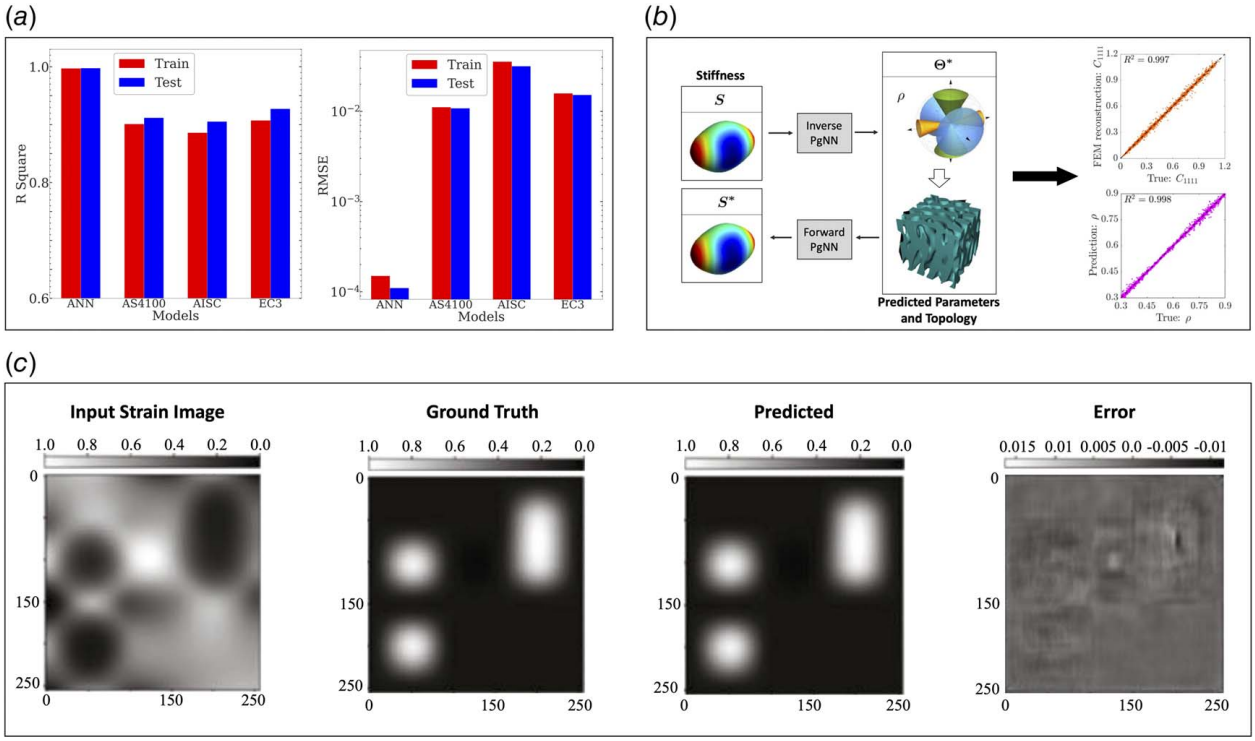

Fig. 4 (*a*) A comparison between ANN and other international codes' accuracy (e.g., R^2 and RMSE) to predict the ultimate moment capacities of castellated beams subjected to lateral-distortional buckling (adapted from the study by Hosseinpour et al. [152]). (*b*) A two-stage PgNN architecture for predicting the design parameters of meta-materials for spindoid topologies. The first ANN (i.e., inverse PgNN) takes the query stiffness as input and the design parameters as output. The second ANN (i.e., forward PgNN) takes the predicted design parameters and reconstructs the stiffness to verify the inverse network accuracy. R^2 values for prediction of stiffness component, C_{1111}, and design parameter, ρ, are shown in the subsets (adapted from the study by Kumar et al. [170]). (*c*) A comparison between conditional GAN and ground truth made by elastography to predict elastic modulus from strain data (adapted from the study by Ni and Gao [171]).

model provided higher accuracy than well-known design codes such as AS4100 [153], AISC [154], and EC3 [155] to model and predict the ultimate moment capacities.

PgNNs have also been utilized in the field of topology optimization for materials and meta-materials [156,157]. Topology optimization is a method that identifies the optimal placement of materials within a defined space to achieve the best structural performance [158,159]. Researchers have developed various PgNN models for this purpose; for instance, Abueidda et al. [160] developed a CNN model that performs real-time topology optimization of linear and nonlinear elastic materials under small and large deformations. The trained model can accurately predict optimal designs without an iterative process and with low inference computation time. Another two-stage approach was proposed by Yu et al. [161], which integrates a CNN-based encoder and decoder in the first stage and a conditional GAN in the second stage to determine near-optimal topological designs. The resulting model can efficiently produce near-optimal structures in terms of pixel values and compliance with significantly reduced computational time. Banga et al. [162] proposed a 3D encoder–decoder CNN for speeding up 3D topology optimization and determining optimal computational strategies for deployment. The proposed model was found to reduce the overall computation time by 40% while achieving an accuracy range of 96%. Lastly, Li et al. [163] presented a GAN-based noniterative near-optimal topology optimizer for conductive heat transfer structures trained on black-and-white density distributions. They combined a GAN for low-resolution topology with a super-resolution generative adversarial network (SRGAN) [164,165], for high-resolution topology solutions in a two-stage hierarchical prediction–refinement pipeline. Compared to conventional topology optimization techniques, this strategy demonstrated clear advantages in terms of computational cost and efficiency.

PgNN has also been applied for inverse design and modeling in solid mechanics. Messner [166] employed a CNN to develop surrogate models that estimate the effective mechanical properties of periodic composites. As an example, the CNN-based model was applied to solve the inverse design problem of finding structures with optimal mechanical properties. The surrogate models were in good agreement with well-established topology optimization methods, such as solid isotropic material with penalization [167], and were sufficiently accurate to recover optimal solutions for topology optimization. Lininger et al. [168] also used CNN to solve an inverse design problem for meta-materials made of thin film stacks. The authors demonstrated the CNN's remarkable ability to explore the large global design space (up to 1012 parameter combinations) and resolve all relationships between meta-material structure and associated ellipsometric and reflectance/transmittance spectra [168,169]. Kumar et al. [170] proposed a two-stage ANN model, as shown in Fig. 4(*b*), for inverse design of meta-materials. The model generates uniform and functionally graded cellular mechanical meta-materials with tailored anisotropic stiffness and density for spinodoid topologies. The ANN model used in this study is a combination of two-stage ANN, first ANN (i.e., inverse PgNN) takes query stiffness as input and output design parameters, e.g., Θ. The second ANN (i.e., forward PgNN) takes the predicted design parameters as input and reconstructs the stiffness to verify the first ANN results. The prediction accuracy for stiffness and the design parameter was validated against ground truth data for both networks; sample comparisons and their corresponding R^2 values are shown in Fig. 4(*b*). Ni and Gao [171] proposed a combination of representative sampling spaces and conditional GAN and cGAN [172,173], to address the inverse problem of modulus identification in the field of elasticity. They showed that the proposed approach can be deployed with high

accuracy, as shown in Fig. 4(*c*) while avoiding the use of costly iterative solvers used in conventional methods, such as the adjoint weighted approach [174]. This model is especially suitable for real-time elastography and high-throughput nondestructive testing techniques used in geological exploration, quality control, and composite material evaluation.

The PgNN models have also been used to overcome some of the computational limitations of multiscale simulations in solid mechanics. This is achieved by (i) bypassing the costly lower-scale calculations and thereby speeding the macro-scale simulations [75], or (ii) replacing a step or the complete simulation with surrogate models [75]. For example, Liang et al. [175] developed an ANN model that takes finite element-based aorta geometry as input and the aortic wall stress distribution as output directly, bypassing FEM calculation. The difference between the stress calculated by FEM and the one estimated by the PgNN model is practically negligible, while the PgNN model produces the output in just a fraction of the FEM computational time. Mozaffar et al. [176] successfully employed RNN-based surrogate models for material modeling by learning the reversible, irreversible, and history-dependent phenomena that occur when studying material plasticity. Mianroodi et al. [2] used a CNN-based solver to predict local stresses in heterogeneous solids with highly nonlinear material response and mechanical contrast features. Compared to common solvers such as FEM, the CNN-based solver offered an acceleration factor of 8300x for elasto-plastic materials. Im et al. [6] proposed a PgNN framework to construct a surrogate model for a high-dimensional elasto-plastic FEM model by integrating an LSTM network with the proper orthogonal decomposition (POD) method [177,178]. The suggested POD-LSTM surrogate model allows rapid, precise, and reliable predictions of elasto-plastic structures based on the provided training dataset exclusively. For the first time, Long et al. [179] used a CNN to estimate the stress intensity factor of planar cracks. Compared to FEM, the key benefit of the proposed lightweight CNN-based crack evaluation methodology is that it can be installed on an unmanned machine to automatically monitor the severity of a crack in real time.

Table 3 reports a nonexhaustive list of recent studies that leveraged PgNNs in solid mechanics and materials design problems. These studies collectively concluded that PgNNs can be successfully integrated with conventional solvers (e.g., FEM solvers) or used as standalone surrogate models to develop accurate and yet faster modeling components for scientific computing in solid mechanics. However, PgNNs come with their own limitations and shortcomings that might compromise solutions under different conditions, as discussed in the next section.

2.3 PgNNs' Limitations. Even though PgNN-based models show great potential to accelerate the modeling of nonlinear phenomena described by input–output interdependencies, they suffer from several critical limitations and shortcomings. Some of these limitations become more pronounced when training datasets are sparse.

- The main limitation of PgNNs is the fact that their training process is based solely on statistics [66]. Even though the training datasets are inherently constrained by physics (e.g., developed by direct numerical simulation, closure laws, and de-noised experimentation), PgNN generates models based on correlations in statistical variations. The outputs (predictions), thus, are naturally physics agnostic [44,189] and may violate the underlying physics [7].
- Another important limitation of PgNNs stems from the fact that training datasets are usually sparse, especially in the scientific fields discussed in this article. When the training data are sparse and does not cover the entire range of underlying physiological attributes, PgNN-based models fail in blind testing on conditions outside the scope of training [50,190], i.e., they do not offer extrapolation capabilities in terms of spatiotemporal variables and/or other physical attributes.
- The predictions of PgNN might be severely compromised, even for inputs within the scope of sparse training datasets [28]. The lack of interpolation capabilities is more pronounced

Table 3 A nonexhaustive list of studies that leveraged PgNNs to model solid mechanics problems

Area of application	NN type	Objective	Reference
Accelerating simulations	ANN	Predicting the aortic wall stress distribution using FEM aorta geometry	[175]
	RNN	Developing surrogate models for material modeling by learning reversible, irreversible, and history-dependent phenomena	[176]
	CNN	Predicting local stresses in heterogeneous solids with the highly nonlinear material response and mechanical contrast features	[2]
	CNN	Estimating stress intensity factor of planar cracks	[179]
Topology optimization	CNN	Optimizing topology of linear and nonlinear elastic materials under large and small deformations	[160]
	CNN-GAN	Determining near-optimal topological design	[161]
	CNN	Accelerating 3D topology optimization	[162]
	GAN-SRGAN	Generating near-optimal topologies for conductive heat transfer structures	[163]
Inverse modeling	CNN	Estimating effective mechanical properties for periodic composites	[166]
	CNN	Solving an inverse design problem for meta-materials made of thin film stacks	[168]
	cGAN	Addressing inverse problem of modulus identification in elasticity	[171]
	CVAE	Designing nano-patterned power splitters for photonic integrated circuits	[171]
Structural elements	ANN	Predicting nonlinear buckling load of an imperfect reticulated shell	[180]
	ANN	Optimizing dynamic behavior of thin-walled laminated cylindrical shells	[181]
	ANN	Determining and identifying loading conditions for shell structures	[151]
	RNN	Identifying and classifying surface defects in industrial steel	[182]
Structural analysis	CNN	Forecasting stress fields in 2D linear elastic cantilevered structures subjected to external static loads	[183]
	ANN	Estimating the thickness and length of reinforced walls based on previous architectural projects	[184]
Condition assessment	Autoencoder-NN	Learning mapping between vibration characteristics and structural damage	[185]
	CNN	Providing a real-time crack assessment method	[186]
	RNN	Nonparametric identification of large civil structures subjected to dynamic loadings	[187]
	CNN	Damage Identification of truss structures using noisy incomplete modal data	[188]

in complex and nonlinear problems where the range of the physicochemical attributes is extremely wide (e.g., the range of Reynolds numbers from creeping flow to turbulent flow).

- PgNNs may not fully satisfy the initial conditions and boundary conditions using which the training datasets are generated [44]. The boundary conditions and computational domain vary from one problem to another, making the data generation and training process prohibitively costly. In addition, a significant portion of scientific computing research involves inverse problems in which unknown physicochemical attributes of interest are estimated from measurements or calculations that are only indirectly related to these attributes [11,15,191,192]. For example, in groundwater flow modeling, we leverage measurements of the pressure of a fluid immersed in an aquifer to estimate the geometry and/or material characteristics of the aquifer [193]. Such requirements further complicate the process of developing a simple neural network that is predictive under any conditions.
- PgNN-based models are not resolution invariant by construction [194], and hence, they cannot be trained on a lower resolution and be directly inferred on a higher resolution. This shortcoming is due to the fact that PgNN is only designed to learn the solution of physical phenomena for a single instance (i.e., inputs–outputs).
- Through the training process, PgNN-based networks learn the input–output interdependencies across the entire dataset. Such a process could potentially consider slight variations in the functional dependencies between different input and output pairs as noise and produce an average solution. Consequently, while these models are optimal with respect to the entire dataset, they may produce suboptimal results in individual cases.
- PgNN models may struggle to learn the underlying process when the training dataset is diverse, i.e., when the interdependencies between different input and output pairs are drastically different [195]. Although this issue can be mitigated by increasing the model size, more data are required to train such a network, making the training costly and, in some cases, impractical.

One way to resolve some of the PgNNs' limitations is to generate more training data. However, this is not always a feasible solution due to the high cost of data acquisition. Alternatively, PgNNs can be further constrained by governing physical laws without any prior assumptions, reducing the need for large datasets. The latter is a plausible solution because, in most cases, the physical phenomenon can be fully and partially described using explicit ODEs, PDEs, and/or closure laws. This approach led to the development of the PiNN [44,51], which is described and reviewed in Sec. 3.

3 Physics-Informed Neural Networks

In scientific computing, physical phenomena are frequently represented mathematically using a strong form consisting of governing differential equations together with initial and boundary conditions. The strong form sets out the constraints that a solution must meet at every point within a domain. The governing equations are often nonlinear or linear PDEs and/or ODEs. Some PDEs, such as the Navier–Stokes equations for fluid flows, and the Föppl–von Kármán equations for solids experiencing large deflections, are particularly challenging to solve [11,196]. There are also other crucial examples of PDEs, including the heat equation [197], wave equation [198], Burgers' equation [199], Laplace's equation [200], and Poisson's equation [201]. This vast body of knowledge can be employed to provide additional constraints on PgNNs while training on available data points [44], if any. To this end, mesh-free PiNNs have been developed [44,51], quickly extended [202–204], and extensively deployed in a variety of scientific and applied fields [205–209]. The readers are referred to the studies by Karniadakis et al. [7] and Cai et al. [11] for the foundational review on how PiNNs function. This section briefly reviews PiNN's core architecture and its state-of-the-art applications in computational fluid and solid mechanics and discusses some of the major limitations.

The PiNN architecture, as depicted in Fig. 5, incorporates the underlying physics outside the neural network to impose known physical laws while training, resulting in outputs that adhere to known physical laws. The widely adopted strategy to address this problem involves the use of a weakly imposed penalty loss to penalize the network if it fails to comply with the physical constraints during training. The PiNN (see Fig. 5) takes spatiotemporal features such as $\mathbf{x}$ and t as input parameters and outputs PDE solution

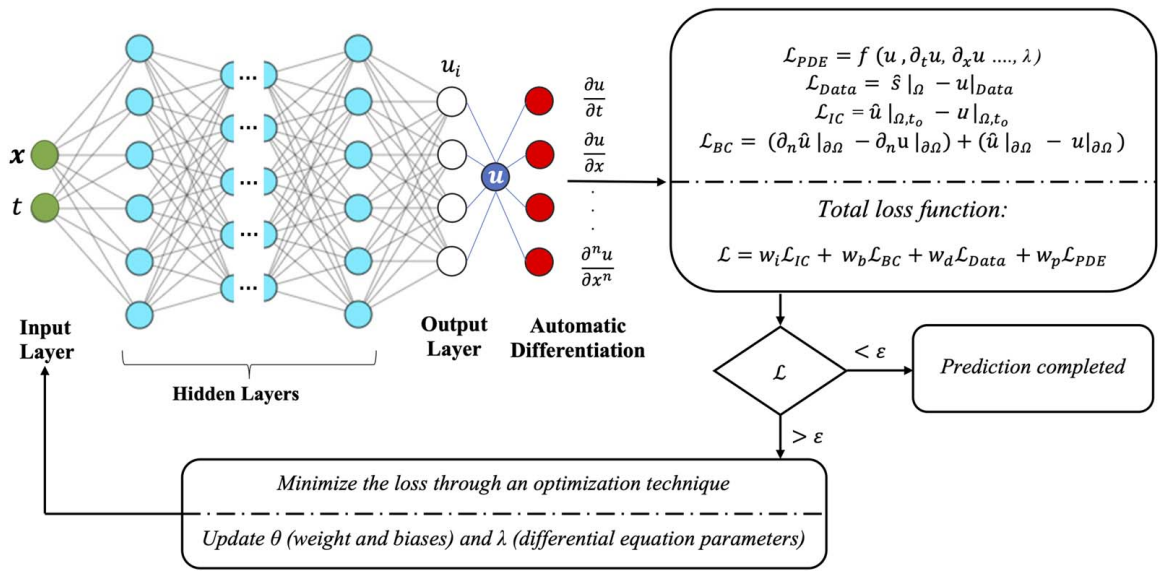

Fig. 5 A schematic architecture of PiNNs. The network digests spatiotemporal coordinates (x,t) as inputs to approximate the multiphysics solution $\hat{u}$. The last layer generates the derivatives of the predicted solution u with respect to the inputs, which are calculated using AD. These derivatives are used to formulate the residuals of the governing equations in the loss function, which is composed of multiple terms weighted by different coefficients. θ and λ are the learnable parameters for weights/biases and unknown PDE parameters, respectively, that can be learned simultaneously while minimizing the loss function.

elements, i.e., **u**, thus making it capable of emulating any PDE. The remaining components of the PiNN architecture, including the hidden layers and activation function, are similar to those of PgNN.

The network's outputs are then fed into the next layer, which is an automated differentiation layer. In this case, multiple partial derivatives are generated by differentiating the outputs with respect to the input parameters (**x** and t). With the goal of optimizing the PDE solution, these partial derivatives are used to generate the required terms in the loss function. The loss function in PiNN is a combination of the loss due to labeled data ($\mathcal{L}_{\text{Data}}$), governing PDEs ($\mathcal{L}_{\text{PDE}}$), applied initial conditions ($\mathcal{L}_{IC}$) and applied boundary conditions ($\mathcal{L}_{\text{BC}}$) [11]. The $\mathcal{L}_{\text{BC}}$ ensures that PiNN's solution meets the specified boundary constraints, whereas $\mathcal{L}_{\text{Data}}$ ensures that the PiNN follows the trend in the training dataset (i.e., historical data, if any). Furthermore, the structure of the PDE is enforced in PiNN through the $\mathcal{L}_{\text{PDE}}$, which specifies the collocation points where the solution to the PDE holds [44]. The weights for the loss due to the initial conditions, the boundary conditions, the data, and the PDE can be specified as w_i, w_b, w_d, and w_p, respectively. The next step is to check, for a given iteration, if the loss is within the accepted tolerance, ϵ. If not, the learnable parameters of the network (θ) and the unknown PDE parameters (λ) are updated through error backpropagation. For a given number of iterations, the entire cycle is repeated until the PiNN model produces learnable parameters with loss functions less than ϵ. Note that the training of PiNNs is more complicated compared to PgNNs, as PiNNs are composed of sophisticated nonconvex and multi-objective loss functions that can result in instability during optimization [7,11,44].

Dissanayake and Phan-Thien [210] were the first to investigate the incorporation of prior knowledge into a neural network. Subsequently, Owhadi [211] introduced the concept of physics-informed learning models as a result of the everincreasing computing power, which enables the use of increasingly complex networks with more learnable parameters and layers. The PiNN, as a new computing paradigm for forward and inverse modeling, was introduced by Raissi et al. in a series of papers [44,51,212]. Raissi et al. [44] deployed two PiNN models, a continuous- and discrete-time model, on examples consisting of different boundary conditions, critical nonlinearities, and complex-valued solutions such as Burgers', Schrodinger's, and Allen-Cahn's equations. The results for Burgers' equation demonstrated that, given a sufficient number of collocation points (i.e., as the basis for the continuous model), an accurate and data-efficient learning procedure can be obtained [44].

In continuous PiNN models, when dealing with higher-dimensional problems, the number of collocation points increases exponentially, making learning processing difficult and computationally expensive [7,44]. Raissi et al. [44] presented a discrete-time model based on the Runge–Kutta technique [213] to address the computational cost issue. This model simply takes a spatial feature as input, and over time-steps, PiNN converges to the underlying physics. For all the examples explored by Raissi et al. [44], continuous and discrete PiNN models were able to satisfactorily build physics-informed surrogate models. Nabian et al. [214] proposed an alternate method for managing collocation points. They investigated the effect of sampling collocation points according to distribution and discovered that it was proportional to the loss function. This concept does not require additional hyperparameters and is simpler to deploy in existing PiNN models. In their study, they claimed that a sampling approach for collocation points enhanced the behavior of the PiNN model during training. The results were validated by deploying the hypothesis on PDEs to solve problems related to elasticity, diffusion, and plane stress physics. Yang et al. [215] proposed Bayesian physics-informed neural network (B-PiNN) for addressing both forward and inverse nonlinear problems governed by PDEs and noisy data. Within this Bayesian framework, a Bayesian neural network is integrated with a PiNN for PDEs to act as the prior, while either Hamiltonian Monte Carlo or variational inference methods are employed as estimators of the posterior. B-PINNs can leverage both the constraints of physical laws and the scattered noisy measurements to provide predictions and quantify the aleatoric uncertainty stemming from the noisy data within the Bayesian framework.

In order to use PiNN to handle inverse problems, the loss function of the deep neural network must satisfy both measured and unknown values at a collection of collocation sites distributed throughout the problem domain. Raissi et al. [51] showed the potential of both continuous and discrete-time PiNN models to solve benchmark inverse problems such as the propagation of nonlinear shallow-water waves (Korteweg–De Vries equation) [216] and incompressible fluid flows (Navier–Stokes equations) [217].

Compared to PgNNs, PiNN models provide more accurate predictions for forward and inverse modeling, particularly in scenarios with high nonlinearities, limited data, or noisy data [218]. As a result, it has been implemented in several fundamental scientific and applied fields. In addition to forward and inverse problems, the PiNN can also be used to develop partial differential equations for unknown phenomena if training data representing the underlying physics of the phenomenon are available [51]. Raissi et al. [51] leveraged both continuous- and discrete-time PiNN models to generate universal PDEs depending on the type and the structure of the available data. In the remainder of this section, we review the recent literature on PiNN's applications in the computational fluid and solid mechanics fields.

3.1 PiNNs for Fluid Mechanics. The application of PiNNs to problems involving fluid flow is an active and ongoing field of study [219,220]. Raissi et al. [212], in a seminal work, proposed the hidden fluid mechanics (HFM) method, which uses PiNNs to encode the governing physical laws of fluid motion, specifically Navier–Stokes equations. By leveraging conservation laws, they extracted hidden variables of interest, such as velocity and pressure fields, from spatiotemporal data visualizations of a passive scalar concentration (such as dye) transported in complex domains. Their PiNN-based algorithm can solve the data assimilation problem without requiring prior knowledge of the geometry, initial, or boundary conditions. The HFM model has demonstrated successful predictions of 2D and 3D velocity and pressure fields in benchmark problems inspired by real-world fluid flow applications. Figure 6, adapted from the study by Raissi et al. [212], compares the prediction of PiNN with the ground truth for the classical problem of a 2D flow past a cylinder. The model can be used to extract valuable quantitative information such as wall-shear stresses and lift and drag forces, for which direct measurements are difficult to obtain.

Zhang et al. [221] also developed a PiNN framework for incompressible fluid flow past a cylinder governed by the Navier–Stokes equations. PiNN learns the relationship between simulation output (i.e., velocity and pressure) and the underlying geometry, boundary, initial conditions, and inherently fluid properties. They showed that generalization performance is enhanced in both the temporal domain and the design space by including Fourier features [222], such as frequency and phase offset parameters. Cheng and Zhang [223] developed Res-PiNN (i.e., Resnet blocks along with PiNN) to simulate cavity flow and flow past a cylinder governed by Burgers' and Navier–Stokes equations. Their results showed that Res-PiNN had better predictive ability than conventional PgNN and vanilla PiNN algorithms. Lou et al. [224] also demonstrated the potential of PiNN to solve inverse multiscale flow problems. They used PiNN for inverse modeling in both the continuum and rare-field regimes represented by the Boltzmann–Bhatnagar-Gross–Krook (BGK) collision model. The results showed that PiNN-BGK is a unified method (i.e., it can be used for forward and inverse modeling), easy to implement, and effective in solving ill-posed inverse problems [224]. Wessels et al. [73] utilized PiNN to develop an updated Lagrangian method for the solution of incompressible free surface flow governed by the inviscid Euler equations, the so-called neural particle method (NPM). The NPM algorithm does not necessitate any specific algorithmic adjustments, which are typically required to accurately address the

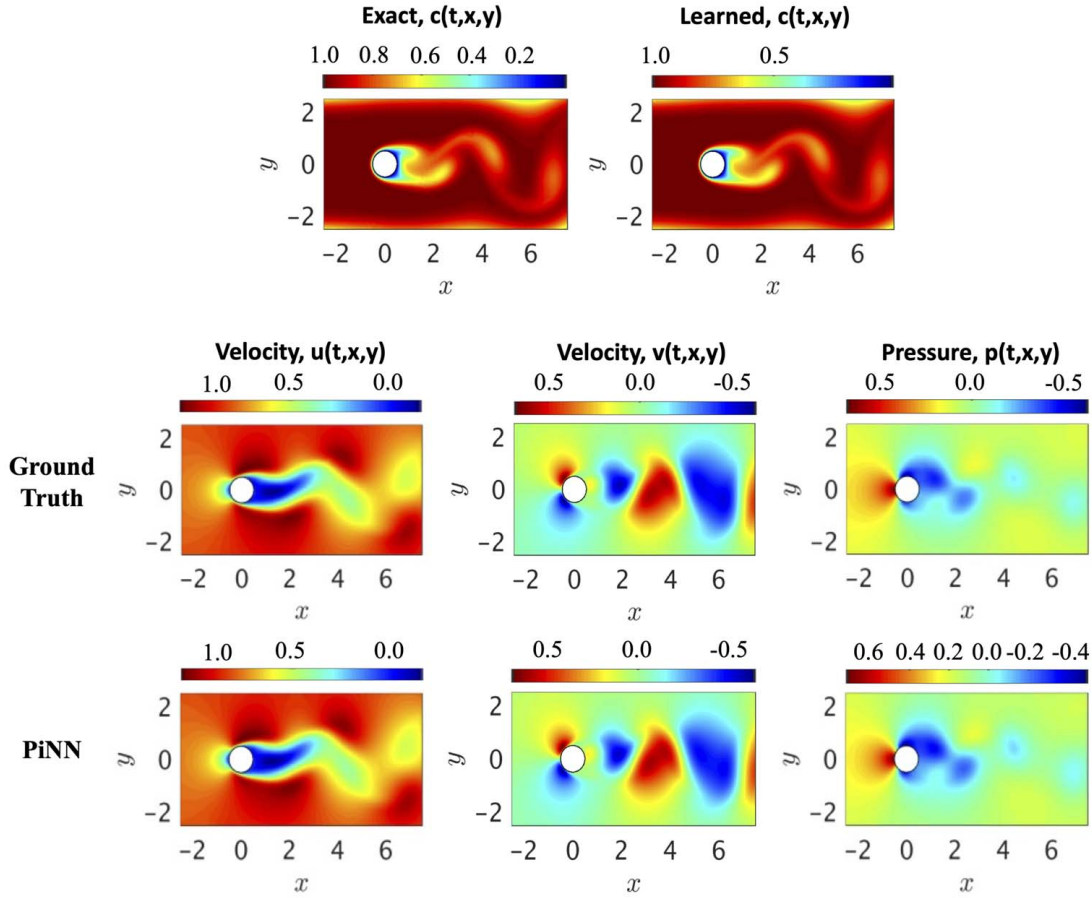

Fig. 6 A comparison between ground truth simulation results and PiNN predictions for a 2D flow past a circular cylinder. Comparisons are shown for the concentration of passive scalar, *c*(*t*, *x*, *y*), and resulting velocity fields, *u* and *v*, and pressure field, *p* (adapted from the study by Raissi et al. [212]).

incompressibility constraint. In their work, it was demonstrated that NPM is capable of accurately computing a pressure field that satisfies the incompressibility condition while avoiding topological constraints in the discretization process [73]. In addition, PiNN has also been employed to model complex non-Newtonian fluid flows involving nonlinear constitutive PDEs able to characterize fluid's rheological behavior [225].

Haghighat et al. [226] trained a PiNN model to solve the dimensionless form of the governing equations of coupled multiphase flow and deformation in porous media. Almajid and Abu-Al-Saud [227] compared the predictions of PiNN with those of PgNN, i.e., a conventional artificial neural network, for solving the gas drainage problem of water-filled porous media. The study showed that PgNN performs well under certain conditions (i.e., when the observed data consist of early and late time saturation profiles), while the PiNN model performs robustly even when the observed data contain only an early time saturation profile (where extrapolations are needed). Depina et al. [228] applied PiNN to model unsaturated groundwater flow problems governed by the Richards PDE and van Genuchten constitutive model [229]. They demonstrated that PiNNs can efficiently estimate the van Genuchten model parameters and solve the inverse problem with a relatively accurate approximation of the solution to the Richards equation.

Some of the other variants of PiNN models employed in fluid mechanics are as follows: *nn-PiNN*, where PiNN is employed to solve constitutive models in conjunction with conservation of mass and momentum for non-Newtonian fluids [225]; *Viscoelastic-Net*, where PiNN is used for stress discovery and viscoelastic flow models selection [230], such as Oldroyd-B [135], Giesekus, and Linear Phan-Thien-Tanner (PTT) [231]; *RhINN,* which is a rheology-informed neural networks employed to solve constitutive equations for a thixotropic-elasto-visco-plastic complex fluid for a series of flow protocols [205]; *CAN-PiNN*, which is a coupled-automatic-numerical differential framework that combines the benefits of numerical differentiation (ND) and AD for robust and efficient training of PiNN [232]; *ModalPiNN*, which is a combination of PiNN with enforced truncated Fourier decomposition [233] for periodic flow reconstruction [234]; *GA-PiNN*, which is a geometry aware PiNN consisted of variational autoencoder, PiNN and boundary constraining network for real-world applications with irregular geometries without parameterization [235]; *Spline-PiNN*, which is a combination of PiNN and hermite spline kernels-based CNN employed to train a PiNN without any precomputed training data and provide fast, continuous solutions that generalize to unseen domains [236]; *conservative PiNN (cPiNN)*, which is a conservative physics-informed neural network consisting of several PiNNs communicating through the subdomain interfaces flux continuity for solving conservation laws [202]; *SA-PiNN*, which is a self-adaptive PiNN to address the adaptive procedures needed to force PiNN to fit accurately the stubborn spots in the solution of stiff PDEs [57]; and *XPiNN*, which is an extended PiNN to enhance the representation and parallelization capacity of PiNN and generalization to any type of PDEs with respect to cPINN [203].

Table 4 reports a nonexhaustive list of recent studies that leveraged PiNN to model fluid flow problems. Furthermore, Table 5 reports a nonexhaustive list of recent studies that developed other variants of PiNN architectures to improve the overall prediction accuracy and computational cost in fluid flow problems.

3.2 PiNNs for Solid Mechanics. The application of PiNNs in computational solid mechanics is also an active field of study

Table 4 A nonexhaustive list of recent studies that leveraged PiNN to model fluid flow problems

Area of application	Objectives	Reference
Incompressible flows	Accelerating the modeling of Navier–Stokes equations to infer the solution for various 2D and 3D flow problems	[51]
	Learning the relationship between output and underlying geometry as well as boundary conditions	[221]
	Simulating ill-posed (e.g., lacking boundary conditions) or inverse laminar and turbulent flow problems	[237]
Turbulent flows	Solving vortex-induced and wake-induced vibration of a cylinder at high Reynolds number	[238]
	Simulating turbulent incompressible flows without using any specific model or making turbulence assumptions	[239]
	Reconstructing Reynolds stress disparities described by Reynolds-averaged Navier–Stokes equations	[240]
Geofluid flows	Solving well-based groundwater flow equations without utilizing labeled data	[241]
	Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport phenomena in porous media	[242]
	Estimating Darcy's law-governed hydraulic conductivity for both saturated and unsaturated flows	[243]
	Solving solute transport problems in homogeneous and heterogeneous porous media governed by the advection–dispersion equation	[56]
	Predicting fluid flow in porous media by sparse observations and physics-informed PointNet	[244]
Non-Newtonian flows	Solving systems of coupled PDEs adopted for non-Newtonian fluid flow modeling	[225]
	Simulating linear viscoelastic flow models such as Oldroyd-B, Giesekus, and Linear PTT	[230]
	Simulating direct and inverse solutions of rheological constitutive models for complex fluids	[205]
Biomedical flows	Enabling the seamless synthesis of noninvasive in vivo measurement techniques and computational flow dynamics models derived from first physical principles	[245]
	Enhancing the quantification of near-wall blood flow and wall-shear stress arterial in diseased arterial flows	[246]
Supersonic flows	Solving inverse supersonic flow problems involving expansion and compression waves	[220]
Surface water flows	Solving ill-posed strongly nonlinear and weakly dispersive surface water waves governed by Serre–Green–Naghdi equations using only data of the free surface elevation and the depth of the water	[247]

Table 5 A nonexhaustive list of different variants of PiNN architectures used in modeling computational fluid flow problems

PiNN structure	Objective	Reference
CAN-PiNN	Providing a PiNN with more accuracy and efficient training by integrating ND- and AD-based approaches	[232]
ModalPiNN	Providing a simpler representation of PiNN for oscillating phenomena to improve performance with respect to sparsity, noise, and lack of synchronization in the data	[234]
GA-PiNN	Enhancing PiNN to develop a parameter-free, irregular geometry-based surrogate model for fluid flow modeling	[235]
Spline-PiNN	Improving the generalization of PiNN by combining it with Hermite splines CNN to solve the incompressible Navier–Stokes equations	[236]
cPiNN	Enhancing PiNN to solve high-dimensional nonlinear conservation laws requiring high computational and memory requirements	[202]
SA-PiNN	Improving the PiNN's convergence and accuracy problem for stiff PDEs using self-adaptive weights in the training	[57]
XPiNN	Improving PiNN and cPiNN in terms of generalization, representation, parallelization capacity, and computational cost	[203]
PiPN	Using point cloud-based neural network to incorporate geometric features of computational domains in PiNN and generalize the solution to unseen domains	[248]
PiGAN	Integrating a physics-informed loss function into SRGAN to address multiphase turbulent fluid flow super-resolution	[249]

[250,251]. The study by Haghighat et al. [252] on modeling linear elasticity using PiNN was among the first papers that introduced PiNN in the solid mechanics community. Since then, the framework has been extended to other solid mechanics problems (e.g., linear and nonlinear elastoplasticity).

Shukla et al. [253] used PiNN for surrogate modeling of the micro-structural properties of poly-crystalline nickel. In their study, in addition to employing the PiNN model, they applied an adaptive activation function to accelerate the convergence of numerical modeling. The resulting PiNN-based surrogate model demonstrated a viable strategy for nondestructive material evaluation. Henkes et al. [254] modeled nonlinear stress and displacement fields induced by inhomogeneities in materials with sharp phase transitions using PiNN. To overcome PiNN convergence issues in this problem, they used adaptive training approaches and domain decomposition [73]. According to their results, the domain decomposition approach is capable of properly resolving nonlinear stress, displacement, and energy in heterogeneous microstructures derived from real-world μCT scans images [254]. Zhang and Gu [255] trained a PiNN model with a loss function based on minimal energy criteria to investigate digital materials. The model tested on 1D tension, 1D bending, and 2D tensile problems demonstrated equivalent performance compared to supervised DL methods (i.e., PgNNs). By adding a hinge loss for the Jacobian matrix, the PiNN method was able to adequately approximate the logarithmic strain and rectify any erroneous deformation gradient.

Rao et al. [256] proposed a PiNN architecture with mixed-variable outputs (displacement and stress components) to handle elastodynamic problems without labeled data. The method was found to boost network's accuracy and trainability in contrast to the pure displacement-based PiNN model. Figure 7 compares the ground truth stress fields generated by the FEM with the ones estimated by mixed-variable PiNN for an elastodynamic problem [256]. It can be observed that stress components can be accurately

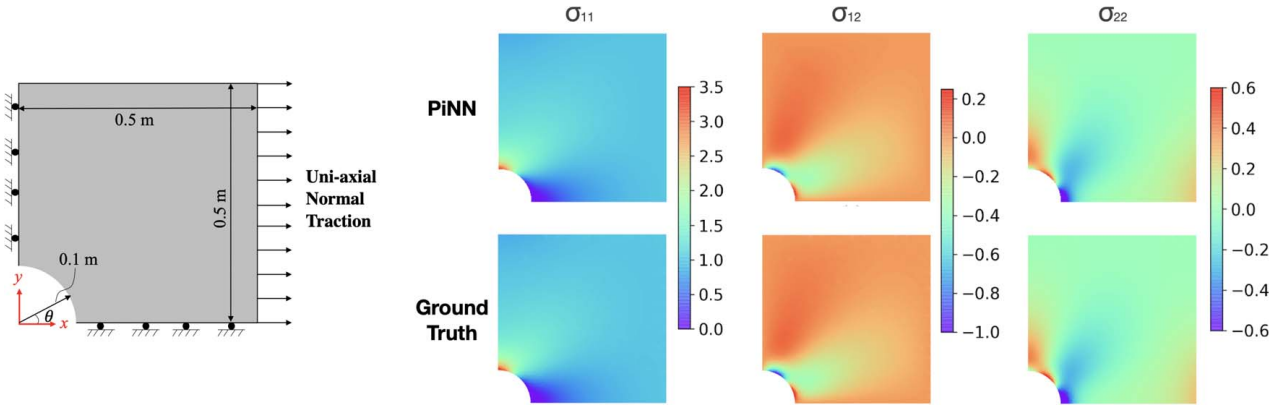

Fig. 7 A comparison between mixed-variable PiNN's prediction and ground truth generated by FEM to predict the stress fields in a defected plate under uniaxial load (adapted from the study by Rao et al. [256]).

estimated by mixed-variable PiNN. Rao et al. [256] also proposed a composite scheme of PiNN to enforce the initial and boundary conditions in a hard manner compared to the conventional (vanilla) PiNN with soft initial and boundary condition enforcement. This model was tested on a series of dynamics problems (e.g., the defected plate under cyclic uniaxial tension and elastic wave propagation) and resulted in the mitigation of inaccuracies near the boundaries encountered by PiNN. Zhou et al. [257] proposed a framework for predicting fatigue life that uses a probabilistic PiNN to incorporate the underlying physics of fatigue. In this framework, a composite loss function is used to enforce physical constraints on derivatives and utilizes a negative log-likelihood function to ensure that the network learns a continuous function that aligns with both experimental data and physics-based insights. Fang and Zhan [258] proposed a PiNN model to design the electromagnetic meta-materials used in various practical applications such as cloaking, rotators, and concentrators. They studied PiNN's inference issues for Maxwell's equation [259] with a high wave number in the frequency domain and improved the activation function to overcome the high wave number problems. The proposed PiNN recovers not only continuous functions but also piecewise functions, which is a new contribution to the application of PiNN in practical problems. Zhang et al. [260] employed PiNN to identify nonhomogenous materials in elastic imaging for application in soft tissues. Two PiNNs were used, one for the approximate solution of the forward problem and the other for approximating the field of unknown material parameters. The results showed that the unknown distribution of mechanical properties can be accurately recovered using PiNN. Abueidda et al. [261] employed PiNN to simulate 3D hyperelasticity problems. They proposed an enhanced PiNN architecture consisting of the residuals of the strong form and the potential energy [262], producing several loss terms that contribute to the definition of the total loss function to be minimized. The enhanced PiNN outperformed both the conventional (vanilla) PiNN and deep energy methods, especially when there were areas of high solution gradients.

Haghighat et al. [15] tested a different variant of PiNN to handle inverse problems and surrogate modeling in solid mechanics. Instead of employing a single neural network, they implemented a PiNN with multiple neural networks in their study. They deployed the framework on linear elastostatic and nonlinear elastoplasticity problems and showed that the improved PiNN model provides a more reliable representation of the physical parameters. In addition, they investigated the domain of transfer learning in PiNN and found that the training phase converges more rapidly when transfer learning is used. Yuan et al. [263] proposed an auxiliary PiNN model (dubbed A-PiNN) to solve inverse problems of nonlinear integro-differential equations (IDEs). A-PiNNs circumvent the limitation of integral discretization by establishing auxiliary output variables in the governing equation to represent the integral(s) and by substituting the integral operator with automated differentiation of the auxiliary output. Therefore, A-PiNN, with its multi-output neural network, is constructed such that it determines both primary and auxiliary outputs to approximate both the variables and integrals in the governing equations. A-PiNNs were used to address the inverse issue of nonlinear IDEs, including the Volterra equation [264]. As demonstrated by their findings, the unknown parameters can be determined satisfactorily even with noisy data.

Some of the other variants of PiNN used in computational solid mechanics are as follows: *PhySRNet*, which is a PiNN-based super-resolution framework for reconstructing high-resolution output fields from low-resolution counterparts without requiring high-resolution labeled data [265]; *peridynamic differential operator (PDDO)-PiNN*, which is a combination of PDDO [266] and PiNN to overcome degrading performance of PiNN under sharp gradients [267]; *PiELM*, which is a combination of PiNN and extreme learning machine (ELM) [268] employed to solve direct problems in linear elasticity [269]; *distributed PiNN (DPiNN)*, which is a distributed PiNN utilizing a piecewise-neural network representation for the underlying field, instead of the piece-polynomial representation commonly used in FEM [58]; and *PiNN-FEM*, which is a mixed formulation based on PiNN and FEM for computational mechanics in heterogeneous domain [270].

Table 6 reports a nonexhaustive list of recent studies that leveraged PiNN in computational solid mechanics. Furthermore, Table 7 reports a nonexhaustive list of recent studies that developed other variants of PiNN architectures to improve overall prediction accuracy and computational cost in solid mechanics modeling.

3.3 PiNNs' Limitations. PiNNs demonstrate significant potential in modeling dynamic systems described by ODEs and PDEs. Nonetheless, they have some limitations and shortcomings that must be considered:

- Vanilla PiNNs use deep networks consisting of a series of fully connected layers and a variant of gradient descent optimization. The learning process and hyperparameter tuning are conducted manually and are sample size and problem dependent. Their training, thus, may face gradient vanishing problems and can be prohibitively slow for practical three-dimensional problems [286]. In addition, vanilla PiNNs impose limitations on low-dimensional spatiotemporal parameterization due to the usage of fully connected layers [46].
- For linear, elliptic, and parabolic PDEs, Shin et al. [287] provided the first convergence theory with respect to the number of training data. They also discussed a set of conditions under which convergence can be guaranteed. However, there is no "solid" theoretical proof of convergence for PiNNs when applied to problems governed by nonlinear PDEs. Note that deep learning models generally fail to realize theoretically

Table 6 A nonexhaustive list of recent studies that leveraged PiNN in computational solid mechanics

Area of application	Objectives	Reference
Elasticity	Solving forward and inverse problems in linear elastostatic and nonlinear elasticity problems	[15]
	Simulating forward and discovery problems for linear elasticity	[252]
	Resolving the nonhomogeneous material identification problem in elasticity imaging	[260]
	Estimating elastic properties of tissues using precompression and postcompression images of objects mimicking properties of tissues	[271]
	Estimating mechanical response of elastic plates under different loading conditions	[272]
	Finding optimal solutions to the reference biharmonic problems of elasticity and elastic plate theory	[273]
	Simulating elastodynamic problems, e.g., elastic wave propagation, deflected plate under periodic uniaxial strain, without labeled data	[256]
Heterogeneous materials	Inferring the spatial variation of compliance coefficients of materials (e.g., speed of the elastic waves) to identify microstructure	[253]
	Resolving nonlinear stress, displacement, and energy fields in heterogeneous microstructures	[254]
	Solving coupled thermomechanics problems in composite materials	[274]
	Predicting the size, shape, and location of the internal structures (e.g., void, inclusion) using linear elasticity, hyperelasticity, and plasticity constitutive models	[275]
Structural elements	Predicting the small-strain response of arbitrarily curved shells	[276]
	Solving mechanical problems of elasticity in one-dimensional elements such as rods and beams	[277]
	Predicting creep-fatigue life of components (316 stainless steel) at elevated temperatures	[278]
	Quantifying physical parameters (e.g., corrosion-fatigue) that are not accounted for in cumulative damage models	[279]
Structural vibrations	Estimating and optimizing vibration characteristics and system properties of structural mechanics and vibration problems	[280]
Digital materials	Resolving physical behaviors of digital materials to design next-generation composites	[255]
Fracture mechanics	Reconstructing the solution of displacement field after damage to predict crack propagation for quasi-brittle materials	[281]
Elasto-viscoplasticity	Modeling the strain rate and temperature dependence of the deformation fields (i.e., displacement, stress, plastic strain)	[282]
Additive manufacturing solid mechanics	Predicting finite fatigue life in materials containing defects	[283]
	Providing a detailed introduction to programming PiNN-based computational solid mechanics from 1D to 3D problems	[284]

Table 7 A nonexhaustive list of different variants of PiNN architectures used in computational solid mechanics problems

PiNN architecture	Objectives	Reference
PhySRNet	Enhancing PiNN using super-resolution techniques to reconstruct downscaled fields from their upscaled counterparts	[265]
Enhanced PiNN	Improving the interpolation and extrapolation ability of PiNN by integrating potential energy, collocation method, and deep learning for hyperelastic problems	[261]
PDDO-PiNN	Improving the performance of PiNN in the presence of a sharp gradient by integrating PDDO and PiNN methods to solve elastoplastic deformation problems	[267]
PiELM	Accelerating PiNN's training process by integrating PiNN and ELM methods to model high-dimensional shell structures	[269]
PiELM	Integrating PiNN and ELM methods to accelerate PiNN's training process in order to model biharmonic equations	[285]
DPiNN	Providing a truly unified framework for addressing problems in solid mechanics solving high-dimensional inverse in heterogeneous media such as linear elasticity	[58]
A-PiNN	Improving PiNN model to solve inverse problems of nonlinear integro-differential equations	[263]
PiNN-FEM	Hybridizing FEM and PiNN to solve heterogeneous media problems such as elasticity and Poisson equation	[270]

established global minima; thus, this limitation is not specific to PiNNs and holds for all deep learning models [7].

- PiNNs contain several terms in the loss function with relative weighting that greatly affects the predicted solution. Currently, there are no guidelines for optimal weight selection [58]. Different terms in the loss function may compete with each other during training, and this competition may reduce the stability of the training process. PiNNs also suffer during training when confronted with an ill-posed optimization problem due to their dependence on soft physical constraints [46].
- PiNNs may face challenges when handling multiscale problems that are governed by low-frequency (e.g., large-scale vortices or polymer relations that vary relatively slowly or smoothly over space or time) or high-frequency (e.g., small-scale turbulence or permeability variation in porous media that vary relatively fast or abruptly over time or space) features [50,288]. The former occurs because low-frequency features necessitate large domains to be entirely resolved, and PiNNs encounter difficulty propagating information from initial or boundary conditions to unobserved areas of a large domain or to future time-steps [289,290].

- PiNNs also face challenges when applied to stiff [291,292] and high-dimensional dynamical systems [293]. Stiff problems may lead to discontinuous solutions, making it difficult for NNs with continuous activation functions to learn [294]. To resolve the issue to some extent, methods such as stiff-PiNNs can be used [295,296]. For high-dimensional problems, the required number of collocation points increases exponentially, making the learning process difficult and computationally expensive [7]. To address this issue to some extent, the importance sampling approach for collocation points [214] can be used.
- PiNNs are solution learning algorithms, i.e., they learn the solutions to a given PDE for a single instance. For any new instance of functional parameters or coefficients, PiNNs require the training of a new neural network [56]. This is because, by construction, PiNNs cannot learn the physical operation of a given phenomenon, and this limits their generalization (e.g., spatiotemporal extrapolation). The PiNN approach thus suffers from the same computational issue as classical solvers, especially in 3D problems (e.g., FEM, FVM), as the optimization problem must be solved for every new instance of PDE parameters, boundary conditions, and initial conditions [64].
- PiNNs encounter difficulties while learning the solutions to inverse problems in heterogeneous media, e.g., a composite slab composed of several materials [286]. In such cases, the parameters of the underlying PDE (e.g., conductivity or permeability coefficients) change across the domain; yet, a PiNN outputs unique parameter values over the whole domain due to its inherent design.

Despite the shortcomings, PiNNs offer a strong promise for complex domains that are hard to mesh and practical problems where data acquisition is expensive. To circumvent some of the limitations of vanilla PiNN, several techniques have been proposed. For instance, to address the first limitation listed earlier, discrete learning techniques using convolutional filters, such as HybridNet [297], dense convolutional encoder–decoder network [298], autoregressive encoder-decoder model [299], TF-Net [300], DiscretizationNet [301], and PhyGeoNet [302], have been employed that exceed vanilla PiNN in terms of computational efficiency. As another example, to address the last limitation listed earlier, Dwivedi et al. [286] proposed a DPiNN that has potential advantages over existing PiNNs to solve the inverse problems in heterogeneous media, which are most likely to be encountered in engineering practices. Some of the other solutions to solve high-dimensional inverse problems are cPiNN [202] and self-adaptive PiNN [57]. Further, XPiNN [203], with its intrinsic parallelization capabilities to deploy multiple neural networks in smaller subdomains, can be used to considerably reduce the computational cost of PiNNs in large (three-dimensional) domains. However, these modifications and alternatives do not solve the generalization problem of PiNNs as the resultant models lack the ability to enforce the existing physical knowledge. To this end, PeNNs have started to emerge. In Sec. 4, we will review the recent literature on physics-encoded neural networks.

4 Physics-Encoded Neural Networks

PeNNs are another family of mesh-free algorithms used in scientific computing, mostly in the fluid mechanics and solid mechanics fields, that strive to *hard-encode underlying physics* (i.e., prior knowledge) into the core architecture of neural networks. Note that, by construction, PeNN-based models extend the learning capability of a neural network from instance learning (imposed by the PgNN and PiNN architectures) to continuous learning [46,60,303]. To hard-encode physical laws (in terms of ODEs, PDEs, closure laws, etc.) into neural network components, different approaches have recently been proposed [9,46,60,194]. PeNN is not a completely new notion, as there has been a long trajectory of research that has proposed the philosophy of building physics constraints constructively into the architectures. For example, one can refer to preserving convexity [304] using deterministic annealing neural network, preserving positivity [305], enforcing symmetries in physics using Lagrangian neural networks (LaNNs) [306,307], capturing trajectories using symplectic recurrent neural networks (SRNNs) [308,309], enforcing exact physics, and extracting structure-preserving surrogate models using data-driven exterior calculus (DDEC) on graphs [310], and imposing different types of physics and probabilistic constraints, including mean values,

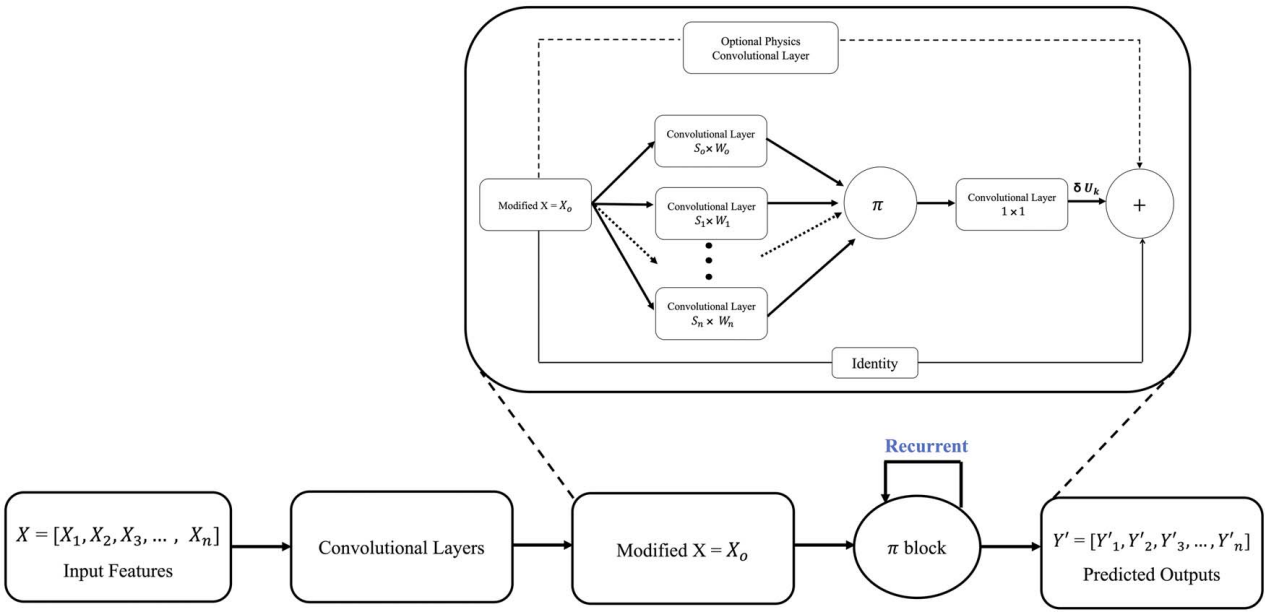

Fig. 8 A schematic architecture for physics-encoded recurrent convolutional neural network [46]. The architecture consists of X as initial inputs, convolutional layers, X_o as full resolution initial state, unconventional convolutional block (π block), and the predicted output layer Y′. Further, the π block consists of multiple parallel convolutional layers whose operations are defined as $S_n \times W_n$, where n is the number of layers; π carries out element-wise product and + carries out element-wise addition.

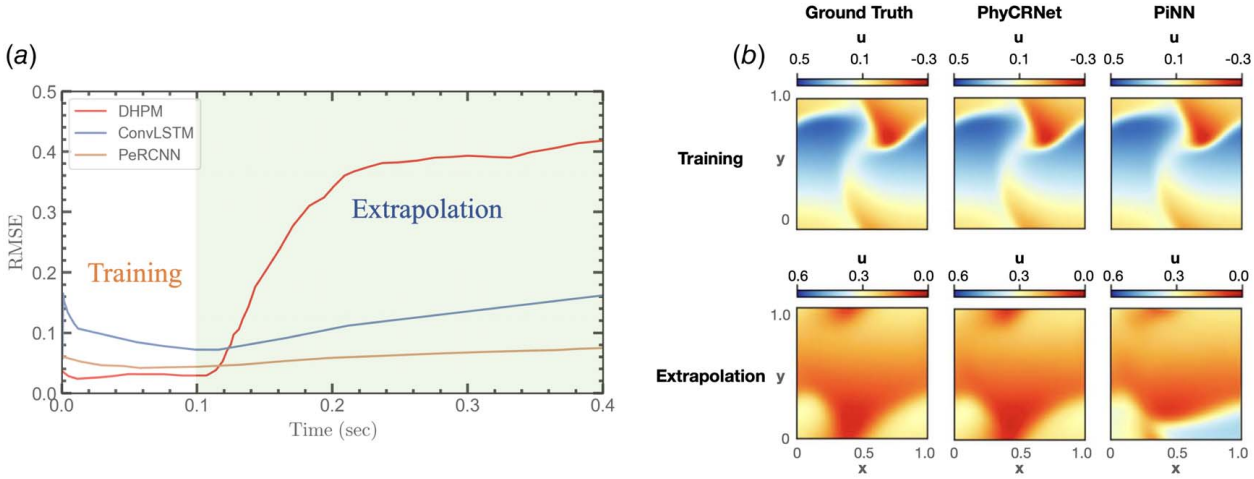

Fig. 9 (*a*) A comparison of error propagation in the training and extrapolation phases for a PeRCNN, deep hidden physics model (DHPM), and a ConvLSTM modeling 2D Burgers' dataset (adapted from the study by Rao et al. [46]). (*b*) A comparison between the predictions by PhyCRNet and PiNN for 2D Burgers' equations. The predicted velocity field in the *x* direction is compared at the training time of $t = 1$ s, and at the extrapolation time of $t = 3$ s (adapted from the study by Ren et al. [316]).

variances, and derivative constraints, on neural network biases and weights for robust curve estimation and uncertainty quantification [311–313]. In this section, we review the two most prominent approaches for encoding physics in neural network architectures and their applications in computational fluid and solid mechanics: (i) physics-encoded recurrent convolutional neural network (PeRCNN) [46,303] and (ii) differential programing (DP) or neural ordinary differential equations (NeuralODEs) [9,60].

4.1 Physics-Encoded Recurrent Convolutional Neural Network. Rao et al. [46] introduced the PeRCNN model, which hard encodes prior knowledge governing nonlinear systems into a neural network. The PeRCNN architecture shown in Fig. 8 facilitates learning in a data-driven manner while forcibly encoding the known physics knowledge. This model exceeds the PgNN and PiNN's capabilities for phenomena in which the explicit formulation of PDEs does not exist and very limited measurement data are available (e.g., Earth or climate system modeling [59]). The encoding mechanism of physics proposed in this model is distinct from the penalty-based physics-informed learning approach and ensures strict adherence to the given physics by the network. The model uses a novel element-wise product operation to achieve nonlinearity, instead of traditional nonlinear activation functions. The results of numerical experiments showed that the physics-encoded learning paradigm produced a highly robust and generalizable model, even in the presence of data noise or scarcity, outperforming some of the existing state-of-the-art data-driven modeling approaches.

As shown in Fig. 8, PeRCNN is made of an input layer, which is constituted by low-resolution noisy initial state measurements $X = [X_1, X_2, X_3, \ldots, X_n]$; a fully convolutional network, as the initial state generator, which downscales/upsamples the low-resolution initial state to a full resolution initial state, dubbed modified X_o, to be used as input to further recurrent computations. For recurrent computing purposes, an unconventional convolutional block, dubbed π, is employed [46]. PeRCNN uses the π block as its main component, which applies multiple convolutional layers to the modified input X_o and combines their feature maps using an element-wise product layer. A 1×1 convolutional layer is added after the product operation to aggregate the output channels. By multiplying the output of the 1×1 convolutional layer by the time spacing δt, the residual of the dynamical system at time t_k can be obtained, which is indicated as δU_k. Ultimately, the last layer generates predictions $Y' = [Y'_1, Y'_2, Y'_3, \ldots, Y'_n]$ by element-wise addition. These operations are shown schematically in Fig. 8.

The PeRCNN architecture was tested on two datasets representing 2D Burgers and 3D Gray–Scott reaction–diffusion equations [46]. In both cases, PeRCNN was compared with convolutional LSTM (ConvLSTM) [314], deep residual network [315], and deep hidden physics models [189] in terms of accuracy (RMSE), data noise/scarcity, and generalization. The comparison for the 2D Burgers' dataset is shown in Fig. 9(*a*), adapted from the study by Rao et al. [46]. The cumulative RMSE for PeRCNN began with a higher value in the training region (due to 10% Gaussian

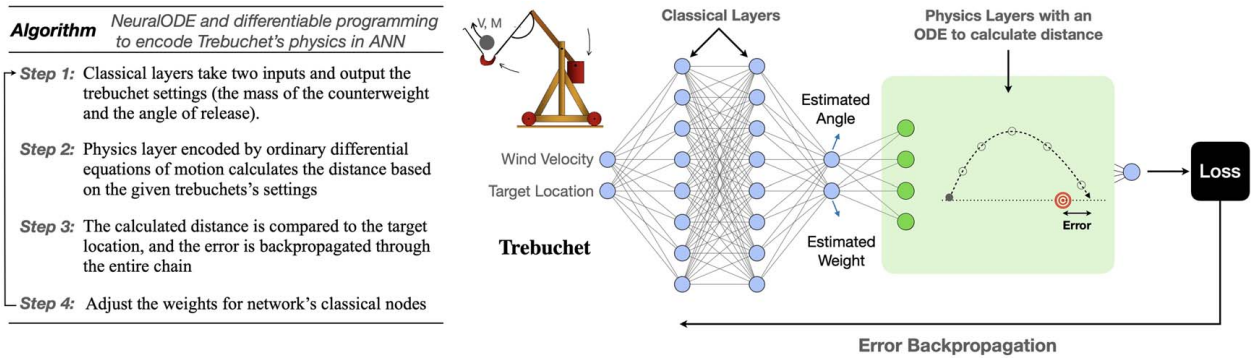

Fig. 10 A NeuralODE architecture that leverages differentiable programing to model the inverse dynamics of a trebuchet. This simple network is 100x faster than direct optimization (adapted from the study by Innes et al. [9]).

noise in the data) and reduced as additional time-steps were assessed. The accumulative RMSE for PeRCNN slightly increases in the extrapolation phase (as a measure of the model's generalization), but clearly surpasses all other algorithms in terms of long-term extrapolation. Rao et al. [303] also used PeRCNN to discover spatiotemporal PDEs from scarce and noisy data and demonstrated its effectiveness and superiority compared to baseline models.

Ren et al. [316] proposed a hybrid algorithm that combined PeRCNN and PiNN to solve the limitations in low-dimensional spatiotemporal parameterization encountered by PgNNs and PiNNs. In the resultant physics-informed convolutional-recurrent network, dubbed PhyCRNet, an encoder–decoder convolutional LSTM network is proposed for low-dimensional spatial feature extraction and temporal evolution learning. In PhyCRNet, the loss function is specified as aggregated discretized PDE residuals. The boundary conditions are hard-coded in the network via designated padding, and the initial conditions are defined as the first input state variable for the network. Autoregressive and residual connections that explicitly simulate time marching were used to enhance the networks. This method ensures generalization to a variety of initial and boundary condition scenarios and yields a well-posed optimization problem in network training. Using PhyCRNet, it is also possible to simultaneously enforce known conservation laws in the network (e.g., mass conservation can be enforced by applying a stream function as the solution variable in the network for fluid dynamics) [316]. Ren et al. [316] evaluated and validated the performance of PhyCRNet using several nonlinear PDEs compared to state-of-the-art baseline algorithms such as the PiNN and autoregressive dense encoder–decoder model [299]. A comparison between PhyCRNet and PiNN to solve Burgers' equation is shown in Fig. 9(*b*) [316]. The results obtained by Ren et al. [316] clearly demonstrated the superiority of the PhyCRNet methodology in terms of solution accuracy, extrapolability, and generalizability.

4.2 Neural Ordinary Differential Equations. The NeuralODE method is another family of PeNN models in which the hidden state of the neural network is transformed from a discrete sequence to a continuous nonlinear function by parametrizing the hidden state derivative using a differentiable function [60]. The output of the network is then computed using a traditional differential equation solver. During training, the error is backpropagated through the network as well as through the ODE solver without access to its internal operations. This architecture is feasible since numerical linear algebra is the common underlying infrastructure for both scientific computing and deep learning, which is bridged by automated differentiation (AD) [317]. Because both differential equations and neural networks are differentiable, standard optimization and error backpropagation techniques can be used to optimize network's weights during training. Instead of learning the nonlinear transformation directly from the training data, the model in NeuralODE learns the structures of the nonlinear transformation. Therefore, because the neural network optimization equations are differentiable, the physical differential equations can be encoded directly into a layer as opposed to adding more layers (e.g., deeper networks). This results in a shallower network mimicking an infinitely deep model that can be continuously inferred at any desired accuracy at reduced memory and computational cost [318].

These continuous-depth models offer features that are lacking in PiNN and PgNNs, such as (i) a reduced number of parameters for supervised learning, (ii) constant memory cost as a function of depth, and (iii) continuous time-series learning (i.e., training with datasets acquired at arbitrary time intervals) [60]. However, the error backpropagation may cause technical difficulties when training such continuous-depth networks. Chen et al. [60] computed gradients using the adjoint sensitivity method [319] while considering the ODE solver as a black box. They demonstrated that this method uses minimal memory can directly control numerical error, and, most importantly, scales linearly with the size of the problem.

Ma et al. [320] compared the performance of the discrete and continuous adjoint sensitivity analysis. They indicated that forward-mode discrete local sensitivity analysis implemented via AD is more efficient than reverse-mode and continuous forward and/or adjoint sensitivity analysis for problems with approximately fewer than 100 parameters. However, in terms of scalability, they showed that the continuous adjoint method is more efficient than the discrete adjoint and forward methods.

Several computational libraries have been implemented to facilitate the practical application of NeuralODE. Poli et al. [321] implemented the TorchDyn library to train NeuralODE models and be as accessible as regular plug-and-play deep learning primitives. Innes et al. [9] and Rackauckas et al. [318] developed GPU-accelerated Zygote and DiffEqFlux libraries in the Julia coding ecosystem to bring differentiable programming and universal differential equation solver capabilities together. As an example, they encoded the ordinary differential equation of motion, as the transformation function, into a neural network to simulate the inverse dynamics of trebuchet [9]. As shown in Fig. 10, the network with classical layers takes the target location and wind speed as input and estimates the weight and angle of the projectile to hit the target. These outputs are fed into the ODE solver to calculate the distance achieved. The model compares the predicted value with the target location and backpropagates the error through the entire chain to adjust the weights of the network. This PeNN model solves trebuchet's inverse dynamics on a personal computer 100x faster than a classical optimization algorithm for this inverse problem. Once trained, this network can be used to aim at any blind target, not just the ones it was trained on; hence, the model is both accelerated and predictive.

NeuralODE has also been integrated with PiNN models, dubbed PiNODE, in order to further constrain the network with known governing physics during training. Such an architecture consists of a neural network whose hidden state is parameterized by an ODE with a loss function similar to PiNN's loss function (see Fig. 5). The loss function penalizes the algorithm based on the data and the strong form of the governing ODEs and backpropagates the error through the application of the adjoint sensitivity methods [320], to update the learnable parameters in the architecture. PiNODE can be deployed to overcome high bias (due to the use of first principles in scientific modeling) and high variance (due to the use of pure data-driven models in scientific modeling) problems. In other words, using PiNODE, prior physics knowledge in terms of ODE is integrated where it is available, and function approximation (e.g., neural networks) is used where it is not available. Lai et al. [322] used PiNODE to model the governing equations in the field of structural dynamics (e.g., free vibration of a 4-degrees-of-freedom dynamical system with cubic nonlinearity). They showed that PiNODE provides an adaptable framework for structural health monitoring (e.g., damage detection) problems. Roehrl et al. [323] tested PiNODE using a forward model of an inverted pendulum on a cart and showed that the approach can learn the nonconservative forces in a real-world physical system with substantial uncertainty.

The application of neural differential equations has also been extended to learn the dynamics of PDE-described systems. Dulny et al. [324] proposed NeuralPDE by combining the method of lines (which represents arbitrarily complex PDEs by a system of ODEs) and NeuralODE through the use of a multilayer convolutional neural network. They tested NeuralPDE on several spatiotemporal datasets generated from the advection–diffusion equation, Burgers' equation, wave propagation equation, climate modeling, and so on. They found that NeuralPDE is competitive with other DL-based approaches, e.g., ResNet [325]. The NeuralPDE's limitations are set by the limitations of the method of lines, e.g., it cannot be used to solve elliptical second-order PDEs. Tables 8 and 9 reports a nonexhaustive list of leading studies that leveraged PeNN to model different scientific problems.

Table 8 A nonexhaustive list of recent studies that leveraged PeNN to model different scientific problems

PeNN structure	Objective	Reference
PeRCNN	Hard-encoding prior knowledge into a neural network to model nonlinear systems	[46]
PhyCRNet	Leveraging the benefits of PeRCNN and PiNN into a single architecture	[316]
NeuralODE	Developing a continuous-depth network by parametrizing the hidden state derivative using a differentiable function	[60]
Zygote/ DiffEqFlux	Providing libraries in the Julia coding ecosystem to facilitate the practical application of NeuralODE.	[9,318]
PiNODE	Integrating PiNN and NeuralODE to provide an adaptable framework for structural health monitoring	[322]
NeuralPDE	Combining the method of lines and NeuralODE to learn the dynamics of PDE-described systems	[324]
LaNN	Capturing symmetries in physical problems such as relativistic particle and double pendulum	[306]
SRNNs	Capturing dynamics of Hamiltonian systems such as the three-body and spring-chain system from observed trajectories	[308]
DDEC	Enforcing exact physics and extracting structure-preserving surrogate models	[310]
Probabilistic NN	Formulating appropriate neural network architecture with weight and bias constraints	[311,312]

Table 9 A nonexhaustive list of recent studies that leveraged neural operators to model different scientific problems

NO structure	Objective	Reference
DeepONets	Learning nonlinear operators accurately and efficiently with low generalization error	[63]
CA-DeepONet	Learning dynamic development of a two-phase mixture and reducing solution time for predicting microstructure evolution	[332]
Pi-DeepONets	Integrating DeepONets and PiNN to relax the requirement for a large training dataset while improving generalization and predictive accuracy	[65]
V-DeepONet	Generating a fast and generalizable surrogate model for brittle fracture mechanics	[333]
FNO	Providing a mesh- and discretization-invariant model to be inferred at any arbitrary spatiotemporal resolutions	[194]
Parallelized FNO	Extending FNO based on domain decomposition to model large-scale three-dimensional problems	[336]
U-FNO	Enhancing the training and testing accuracy of FNO for large-scale, highly heterogeneous geological formations	[338]
IFNO	Capturing the long-range dependencies in the feature space that yields an implicit neural operator to model the complex responses of materials due to their heterogeneity and defects	[339]
PINO	Integrating PiNN and FNO to relax the requirement for a large training dataset while enhancing generalization and optimization	[50]

4.3 PeNNs' Limitations. Despite the advancement of numerous PeNN models and their success in modeling complex physical systems, these new architectures also face several challenges. The most important one is attributed to training. PeNN-based models promote continuous learning using the development of continuous-depth networks, making PeNNs more difficult to train than PgNNs and PiNNs. Considering this, most of the limitations faced by PgNN and PiNN (e.g., convergence rate, stability, scalability, sample size dependency, and problem dependency) are also faced by PeNN. In addition, PeNNs usually have complex architectures, and their implementation is not as straightforward as PiNNs or PgNNs. In spite of PeNNs' implementation complexity, their efficient algorithms in the finite-dimensional setting, their ability to provide transferable solutions, their robustness against data scarcity, and their generalizability compared to PgNN and PiNN make them have a great potential to significantly accelerate traditional scientific computing for applications in computational fluid and solid mechanics.

5 Neural Operators

Most of the scientific deep learning methods discussed so far, e.g., PgNNs, PiNNs, and PeNNs, are generally designed to map the solution of a physical phenomenon for a single instance (e.g., a certain spatiotemporal domain and boundary conditions to solve a PDE using PiNN) and thus must be retrained or further trained (e.g., transfer learning [326]) to map the solution under a different instant. Another way to alleviate this problem is to use neural operators that learn nonlinear mappings between function spaces [45,327,328]. Neural operators, thus, form another simulation paradigm that learns the underlying linear and nonlinear continuous operators using advanced architecture. These models, similar to PgNNs, enforce the physics of the problem using labeled input–output dataset pairs, but provide enhanced generalization, interpretability, continuous learning, and computational efficiency compared to PgNNs as well as PiNNs and PeNNs [50,60,194].

This new paradigm uses mesh-invariant, infinite-dimensional operators based on neural networks that do not require a prior understanding of PDEs. Neural operators work with the data to learn the resolution-invariant solution to the problem of interest [50]. In other words, neural operators can be trained on one spatiotemporal resolution and successfully inferred on any other [328]. This resolution-invariant feature is achieved using the fact that a neural operator learns continuous functions rather than discretized vectors, by parameterizing the model in function spaces [50,328]. Note that PgNNs and PiNNs, for example, using MLP, may also guarantee a small generalization error, but that is only achieved by sufficiently large networks. One distinct feature of neural operators is their robustness for applications that require real-time inference, especially when combined with PiNN [64,65]. Three main neural operators have recently been proposed, namely, (i) DeepONets [63], (ii) Fourier neural operator (FNO) [194], and (iii) graph neural operator (GNO) [328,329]. A recent review by Goswami et al. [64] extensively compared these neural operators. In this section, we briefly review DeepONets and FNO as the two prominent neural operators that can be applied in computational fluid and solid mechanics.

5.1 Deep Operator Networks. Lu et al. [45] developed DeepONets based on the universal approximation theorem for operators [330] that can be used to learn operators accurately and efficiently with very small generalization errors. Lu et al. [63] proposed two architectures known as stacked and unstacked for DeepONet. The stacked DeepONet architecture is shown in Fig. 11(*a*), which consists of one trunk network and multiple stacked branch

networks, $k = 1, 2 \ldots, p$. The stacked DeepONet is formed by selecting the trunk network as a one-layer network with width p and each branch network as a one hidden layer network with width n. To learn an operator $G{:}s \rightarrow G(s)$, the stacked DeepONet architecture takes the function s as input to branch networks and y (i.e., points in the domain of $G(s)$) as input to the trunk network. Here, the vector $[(x_1), (x_2), \ldots, (x_m)]$ represents the finite data locations, alternatively dubbed sensors. The trunk network outputs $[t_1, t_2, \ldots, t_p]^T \in \mathbb{R}^p$, and each branch network outputs a scalar represented by $b_k \in \mathbb{R}$, where $k = 1, 2 \cdots, p$. Next, the outputs generated by trunk and branch networks are integrated together as $G(s)(y) \approx \sum_{k=1}^{p} b_k(s(x_1), s(x_2), \ldots s(x_m))t_k(y)$. The unstacked DeepONet architecture is also shown in Fig. 11(*a*), which consists of only one branch network (depicted in dark blue) and one trunk network. Unstacked DeepONet can be considered as a stacked DeepONet, in which all branch networks share the same set of parameters [63]. DeepONet was first used to learn several explicit operators, including integral and fractional Laplacians, along with implicit operators that represented deterministic and stochastic differential equations [63]. The two main advantages of DeepONets discussed by Lu et al. [63] are as follows: (i) small generalization error and (ii) rapid convergence of training and testing errors with respect to the quantity of training data.

Lin et al. [331] showed the effectiveness of DeepONet against data density and placement, which is advantageous when no prior knowledge is available on how much training data are required or when there are strict limits on data acquisition (e.g., location accessibility). To this end, they employed DeepONet and LSTM (i.e., PgNN) to model datasets that represent the formation of a single bubble in response to time-varying changes in ambient liquid pressure. To generate datasets, they used Rayleigh–Plesset (R–P) as a macroscopic model and dissipative particle dynamics as a microscopic model. They used Gaussian random fields (GRFs) to generate different pressure fields, which serve as input signals for this dynamical system. The results of the comparison are shown in Fig. 11(*b*). The top row shows the prediction results for the liquid pressure trajectory when only 20 data points are known per trajectory, i.e., sparse training data, and the bottom row shows the same but when 200 data points are known per trajectory, i.e., dense training data. As shown, regardless of how sparse the training data was, DeepONet was able to outperform LSTM in predicting the liquid pressure trajectory.

In addition, they examined a case where the input was not contained within the training input range, i.e., when the correlation length of the pressure field was outside of the training range. In this case, they were initially unable to make accurate predictions, but mitigated the issue by transferring learning to a pretrained DeepONet trunk network and fine-tuning it with only a few additional data points. They also demonstrated that DeepONet can learn the mean component of the noisy raw data for the microscopic model without any additional data processing and that the computational time can be reduced from 48 CPU hours to a fraction of a second. These results confirmed that the DeepONet model can be applied across macroscopic and microscopic regimes of bubble growth dynamics, establishing the foundation for a unified neural network model that can seamlessly predict physics interacting across scales.

Oommen et al. [332] proposed a method called convolutional autoencoder (CA)-DeepONet, which combines CA with DeepONet to accelerate the prediction of microstructure evolution by learning the dynamic development of a two-phase mixture. The CA-DeepONet approach uses the low-dimensional latent space of the CA to obtain a compressed representation of the microstructure data, while the DeepONet is used to learn the mesoscale dynamics of microstructure evolution from the latent space of the autoencoder. The decoder component of the CA then reconstructs the evolution of the microstructure based on DeepONet's predictions. The trained DeepOnet architecture can then be used to speed up the numerical solver in extrapolation tasks or substitute the high-fidelity phase-field numerical solver in interpolation problems. By taking inspiration from PiNNs for sparse data domains, DeepONets can also be trained with very sparse labeled datasets while incorporating known differential equations into the loss function. This approach results in physics-informed DeepONets (Pi-DeepONets) [65,333]. Wang et al. [65] employed Pi-DeepONets for benchmark problems such as diffusion reaction, Burger's equation, advection equation, and eikonal equation. Compared to vanilla DeepONet, the result reveals significant improvements in predictive accuracy, generalization performance, and data efficiency. Furthermore, Pi-DeepONets can learn a solution operator without any paired input–output

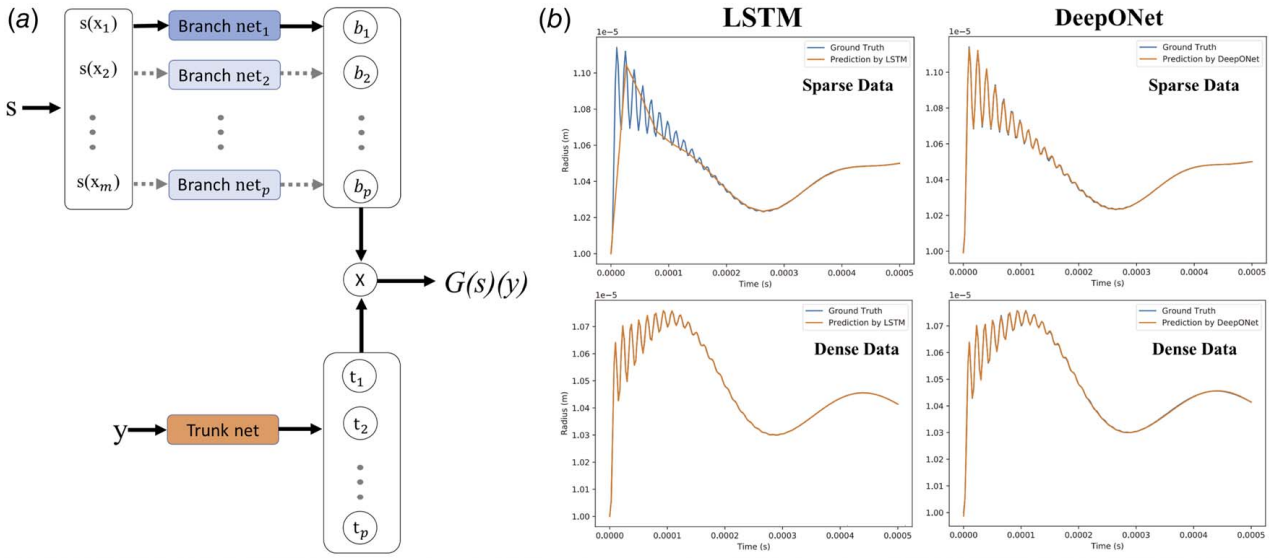

Fig. 11 DeepONet for computational mechanics. (*a*) The general architecture of stacked DeepONets to learn an operator $G{:}s \rightarrow G(s)$. The stacked DeepONet reduces to an unstacked DeepONet when only one branch network is used. Alternatively, the unstacked DeepONet may be considered as a stacked DeepONet, in which all of the branch networks share the same set of parameters [63]. (*b*) A comparison between DeepONet and LSTM (i.e., PgNN) to model sparse and dense datasets representing the formation of a single bubble in response to time-varying changes in the ambient liquid pressure (adapted from the study by Lin et al. [331]).

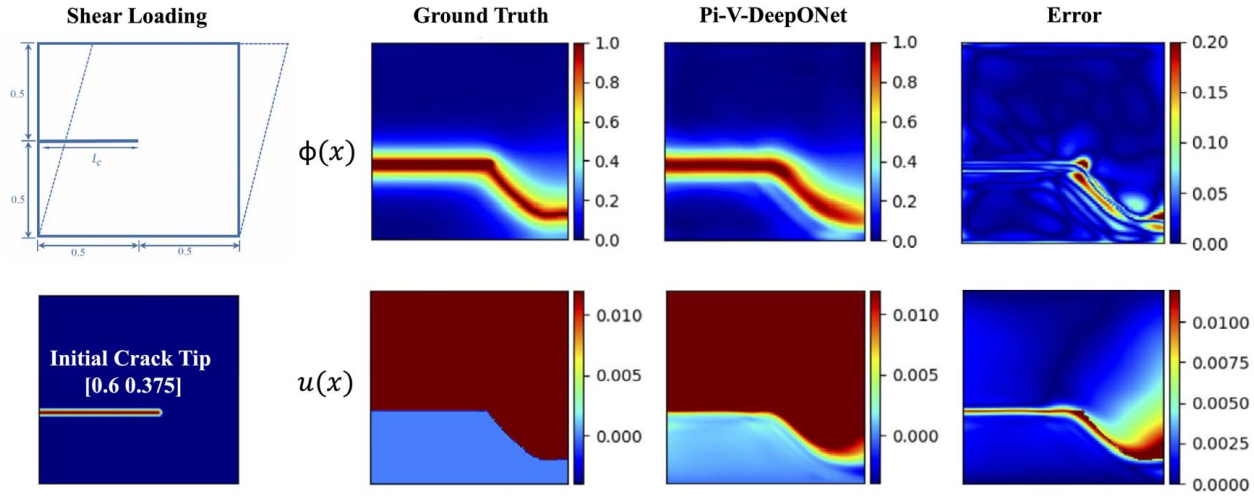

Fig. 12 A comparison between Pi-V-DeepONet and isogeometric FEM analysis to predict the final damage path for the shear failure of a single-edge notched plate. The initial geometry of the single-edge notched plate, shear loading, and the initial configuration of the crack are shown on the left panels. The right panels show the comparisons for the phase field, $\phi(x)$, and displacement along the x-axis, $u(x)$, as well as the corresponding error between the prediction and ground truth (adapted from the study by Goswami et al. [333]).

training data, allowing them to simulate nonlinear and nonequilibrium processes in computational mechanics up to three orders of magnitude faster than traditional solvers [65].

Goswami et al. [333] used a physics-informed variational formulation of DeepONet (Pi-V-DeepONet) for the mechanics of brittle fractures. Training of the Pi-V-DeepONet was performed using the governing equations in a variational form and some labeled data. They used the Pi-V-DeepONet framework to determine failure pathways, failure zones, and damage along failure in brittle fractures for quasi-brittle materials. They trained the model to map the initial configuration of a defect (e.g., crack) to the relevant fields of interest (e.g., damage and displacements, see Fig. 12). They showed that their model can rapidly predict the solution to any initial crack configuration and loading steps. In brittle fracture mechanics, the proposed model can be used to improve design, assess reliability, and quantify uncertainty.

Due to the high cost of evaluating integral operators, DeepONets may face difficulty in developing effective numerical algorithms capable of replacing convolutional or recurrent neural networks in an infinite-dimensional context. Li et al. [194] made an effort along this line and developed an operator regression by parameterizing the integral kernel in the Fourier space and termed it the FNO. In Sec. 5.2, we discuss the core architecture of FNO and the recent developments around it.

5.2 Fourier Neural Operator. In order to benefit from neural operators in infinite-dimensional spaces, Li et al. [194] developed a neural operator in the Fourier space, dubbed FNO, with a core architecture schematically shown in Fig. 13. Training starts with an input X, which is subsequently elevated to a higher-dimensional space by a neural network S. The second phase involves the use of several Fourier layers of integral operators and activation functions. In each Fourier layer, the input is transformed using (i) a Fourier transform, F; (ii) a linear transform, T, in the lower Fourier modes that filters out the higher modes; and (iii) an inverse Fourier transform, F^{-1}. The input is also transformed using a local linear transform, W, before the application of the activation function, σ. The Fourier layers are designed to be discretization invariant since they learn from functions that are discretized arbitrarily. Indeed, the integral operator is applied in convolution and is represented as a linear transformation in the Fourier domain, allowing the FNO to learn the mapping over infinite-dimensional spaces. The result of the Fourier layer is projected back to the target dimension in the third phase using another neural network M, which eventually outputs the desired output Y' [194]. Unlike other DL methods, the FNO model error is consistent regardless of the input and output resolutions (e.g., in PgNN methods, the error grows with the resolution).

Li et al. [194] employed FNO on three different test cases, including the 1D Burgers' equation, the 2D Darcy flow equation, and the

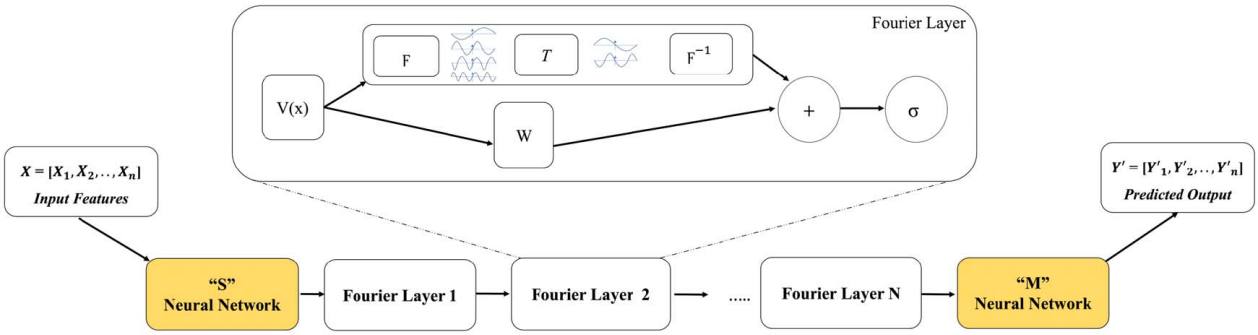

Fig. 13 A schematic architecture of the FNO [194]. Here, X is the input, S and M are the neural network for dimensional space operation, and Y′ is the predicted outputs. Each of the Fourier layers consists of v(x) as the initial state, $\mathcal{F}$ layer that performs Fourier transform, T layer that performs linear transform, $\mathcal{F}^{-1}$ layer that performs inverse Fourier transform, W layer for local linear transform, + block that carries out element-wise addition on W, and the final output from the $\mathcal{F}^{-1}$ layer. The final product from element-wise addition is passed on to a σ layer.

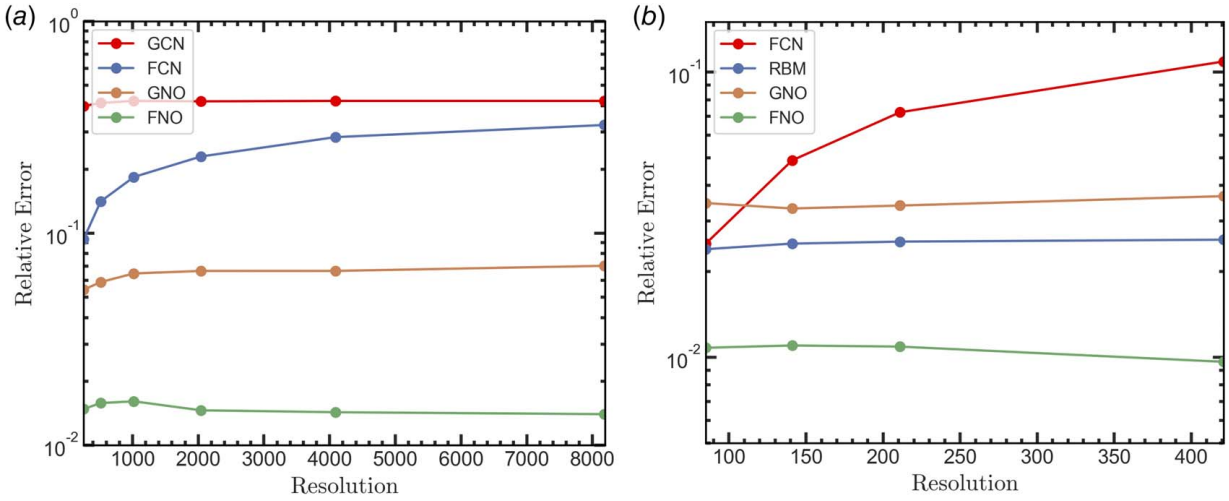

Fig. 14 Error comparison between FNO and other state-of-the-art methods (reduced bias method (RBM), fully convolutional network (FCN), graph neural operator (GNO), and graph convolutional network (GCN)) for (*a*) Burgers' equation and (*b*) Darcy's flow equation at different resolutions (adapted from the study by Li et al. [194])

2D Navier–Stokes equations. For each test case, FNO was compared with state-of-the-art models. In particular, for Burgers' and Darcy's test cases, the methods used for comparison were the conventional ANN (i.e., PgNN), reduced bias method [334], fully convolutional networks [335], principal component analysis as an encoder in the neural network [327], graph neural operator [328], and low-rank decomposition neural operator (i.e., unstacked DeepONet [45]). In all test cases, FNO yielded the lowest relative error. The model error comparisons for 1D Burgers' and 2D Darcy flow equations are depicted in Fig. 14, adapted from the study by Li et al. [194].

The FNO model, as stated, can be trained on a specific resolution and tested on a different resolution. Li et al. [194] demonstrated this claim by training a FNO on the Navier–Stokes equations for a 2D test case with a resolution of $64 \times 64 \times 20$ (n_x, n_y, n_t) standing for spatial (x, y) and time resolution, and then evaluating it with a resolution of $256 \times 256 \times 80$, as shown in Fig. 15(*a*). Compared to other models, the FNO model was the only technique capable of

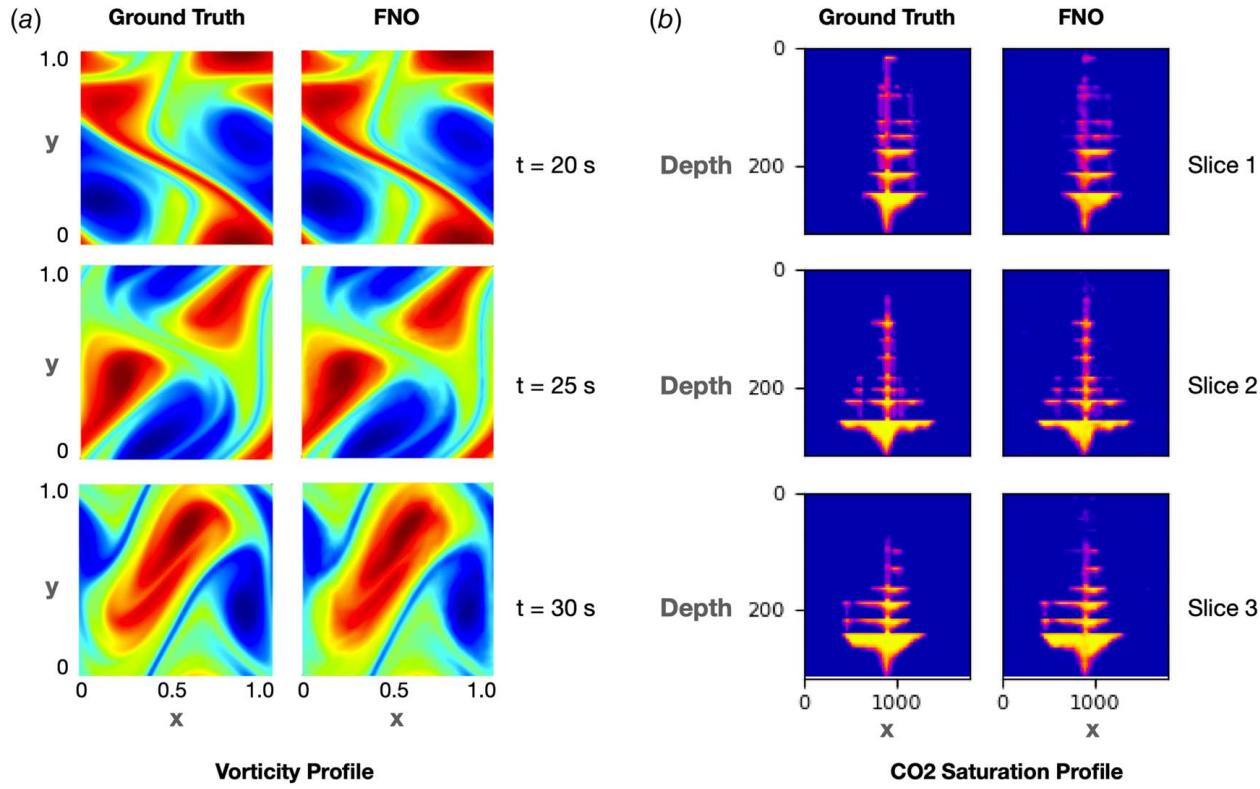

Fig. 15 A qualitative comparison between predictions made by the Fourier neural operator (FNO) and ground truth values. (*a*) The comparison for a FNO model trained on $64 \times 64 \times 20$ resolution (n_x, n_y, n_t) and evaluated on $256 \times 256 \times 80$ resolution to solve 2D Navier–Stokes equations (adapted from the study by Li et al. [194]). (*b*) The comparison for a parallelized FNO model trained to solve large-scale 3D subsurface CO_2 flow modeling evaluated at $60 \times 60 \times 64 \times n_t$ resolution (adapted from the study by Grady et al. [336]).

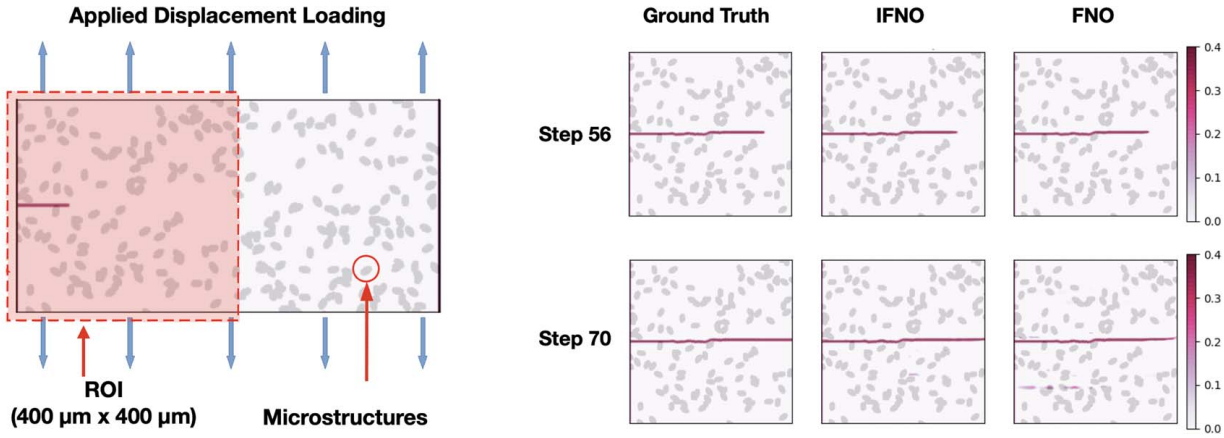

Fig. 16 A comparison between the IFNO and FNO models to predict crack propagation and damage field within a region of interest (ROI) in a pre-cracked glass–ceramics experiment with randomly distributed material property fields (adapted from the study by You et al. [339])

performing resolution downscaling both spatially and temporally [194]. FNOs can also achieve several orders of magnitude speedup factors over conventional numerical PDE solvers. However, they have only been used for 2D or small 3D problems due to the large dimensionality of their input data, which increases in the number of network weights significantly. With this problem in mind, Grady et al. [336] proposed a parallelized version of FNO based on domain-decomposition to resolve this limitation. By using this extension, they were able to use FNO in large-scale modeling, e.g., simulating the transient evolution of the CO_2 plume in subsurface heterogeneous reservoirs as part of the carbon capture and storage technology [337], see Fig. 15(*b*). The input to the network (with a similar architecture to that proposed by Li et al. [194]) was designed to be a tensor containing both the permeability and topography fields at each 3D spatial position using a $60 \times 60 \times 64$ (n_x, n_y, n_z) resolution, and the output was $60 \times 60 \times 64 \times n_t$. For a time resolution of $n_t = 30$ s, they found that the parallelized FNO model was 271 times faster (without even leveraging GPU) than the conventional porous media solver while achieving comparable accuracy. Wen et al. [338] proposed U-FNO, an extended version of FNO, to simulate multiphase flows in porous media. Their study focused on the CO_2–water multiphase flow through a heterogeneous medium under a wide range of reservoir conditions, injection configurations, flowrates, and multiphase flow properties. U-FNO was compared to FNO and CNN (i.e., PgNN) and found to perform better in predicting gas saturation and pressure build up in highly heterogeneous geological formations. However, the authors noted that while U-FNO improved the training accuracy of FNO, it did not inherently allow flexibility in training and testing at different discretizations.

You et al. [339] proposed implicit Fourier neural operator (IFNO) to model the behavior of materials with defects and heterogeneity without relying on traditional constitutive models. As the depth of the network increases, the IFNO model can represent a fixed-point equation that generates an implicit neural operator and can capture long-range dependencies within the feature space (e.g., it can mimic displacement/damage fields). You et al. [339] demonstrated the performance of IFNO using a series of test cases such as hyperelastic, anisotropic, and brittle materials. Figure 16 shows a comparison between IFNO and FNO for the transient propagation of a glass-ceramic crack [339]. As demonstrated, IFNO outperforms FNO (in terms of accuracy) and conventional constitutive models (in terms of computational cost) to predict the displacement field.

The FNO model has also been hybridized with PiNN to create the so-called physics-informed neural operator (PiNO) [50]. The PiNO framework is a combination of operating-learning (i.e., FNO) and function-optimization (i.e., PiNN) frameworks that improves convergence rates and accuracy over both PiNN and FNO models. This integration was suggested to address the challenges in PiNN (e.g., generalization and optimization, especially for multiscale dynamical systems) and the challenges in FNO (e.g., the need for expensive and impractical large training datasets) [50]. Li et al. [50] deployed the PiNO model on several benchmark problems (e.g., Kolmogorov flow, lid-cavity flow) to show that PiNO can outperform PiNN and FNO models while maintaining the FNO's exceptional speedup factor over other solvers.

5.3 NOs' Limitations. DeepONet [45] and FNO [194], as the two most common neural operators to date, share some commonalities but also have significant differences. The DeepONet architecture was developed based on the universal approximation theorems proposed by Chen and Chen [330], while the FNO architecture was based on parameterizing the integral kernel in Fourier space. However, the continuous form of FNO can be seen as DeepONet with a specific trunk architecture (using a trigonometric basis) and branch networks [340]. Unlike DeepONet, FNO discretizes both the input and output functions via point-wise evaluations in an evenly spaced mesh. As a result, after network training, FNO can only predict the solution in the same mesh as the input function, whereas DeepONet can make predictions at any arbitrary location. In addition, FNO requires a full field of observation data for training, while DeepONet is more flexible in this regard, except POD-DeepONet [341] that requires a full field of observation data to calculate POD modes. DeepONet, FNO, and their various variants still face some limitations, especially when applied to large multi-physics problems, that necessitate further investigations.

- Neural operators are purely data driven and require relatively large training datasets; therefore, they face constraints when applied to problems where data acquisition is complex and/or costly [341]. Integration with PiNN can resolve this issue to some extent for problems where the underlying physics is fully known and can be integrated into the loss function [65]. Also, for practical applications, training NOs based solely on the governing equations in the loss function can produce inaccurate predictions; instead, hybrid physics-data training is recommended [65].
- Neural operators encounter challenges when learning from diverse datasets, especially when the input functions are sampled from various GRFs with different length scales [342]. To address this problem, one can analyze the extrapolation behavior of neural operators by measuring the complexity of the extrapolation with the 2-Wasserstein distance [343] and taking into account the bias–variance trade-off for extrapolation with respect to the capacity of the model [344].

- Neural operators, especially DeepONets, can face limitations when dealing with large time domains due to high dimensionality, leading to training challenges and overfitting [345]. To tackle this limitation, one can restructure DeepONet learning so that it recursively learns the solution trajectories of dynamical systems with time-dependent inputs [346].
- DeepONet and FNO are typically limited to basic geometry and/or structured data (e.g., 2D or small 3D problems) due to the large dimensionality of their input data that significantly increases the number of network weights) [341]. They are also prone to overfitting as the number of trainable parameters increases, making the training process more difficult [347]. IFNOs have addressed this challenge to some extent [339]. In IFNO, the solution operator is first formulated as an implicitly defined mapping and then modeled as a fixed point. The latter aims to overcome the challenge of network training in the case of deep layers, and the former minimizes the number of trainable parameters and memory costs. However, due to the finite size of the neural network architecture, the convergence of NOs (e.g., DeepONet) error for the size of the training data becomes algebraic for large datasets; it is desired to be exponential [341].
- FNO might not be reliable for discontinuous functions as it relies on the Fourier transformation. This is mitigated to some extent by DeepONet, as it was shown to perform well for functions with discontinuity (e.g., compressible Euler equations) [341].

Despite these limitations, neural operators are the leading algorithms in a variety of real-time inference applications (in vanilla form [63,65,344] or combined with PiNN [64]), including autonomous systems, surrogates in design problems, and uncertainty quantification. For the latter, the variational Bayes DeepONet (VB-DeepONet) was introduced to provide a Bayesian operator learning framework that improves the robustness and generalization capabilities of DeepONets while providing essential predictive uncertainty information [348]. Yang et al. [349] proposed an approach to quantify uncertainty in DeepONets, showing its efficacy in improving prediction robustness, estimating uncertainty in data-scarce scenarios, and identifying out-of-distribution examples. Additionally, they introduced a specialized library, UQDeepONet, designed to facilitate large-scale applications of uncertainty quantification in DeepONets. The significance of uncertainty quantification in neural operators lies in its ability to provide methodologies and tools to meet the increasing demand for dependable and interpretable uncertainty quantification when integrating neural networks with mathematical principles in scientific and engineering applications. Lin et al. [350] also proposed a Bayesian DeepONet, which leverages replica exchange Langevin diffusion to handle noisy data. They showed that Bayesian DeepONet, compared to DeepONets trained with gradient-based algorithms, significantly improves training convergence and provides accurate uncertainty estimates for noisy scenarios.

6 Conclusions and Future Research Directions

A considerable number of research topics collectively support the efficacy of combining scientific computing and deep learning approaches. In particular, this combination improves the efficiency of both forward and inverse modeling for high-dimensional problems that are prohibitively expensive, contain noisy data, require a complex mesh, and are governed by nonlinear, ill-posed differential equations. The everincreasing computer power will continue to further furnish this combination by allowing the use of deeper neural networks and considering higher-dimensional interdependencies and design space. This can result in enhancing predictive capabilities for fluid and solid mechanics problems and discovering new physics and hidden patterns governing fluid and solid behaviors.

Table 10 A comparison of the main characteristics of PgNNs, PiNNs, PeNNs, and NOs to model nonlinear multiscale phenomena in computational fluid and solid mechanics

Feature	PgNNs	PiNNs	PeNNs	NOs
Accelerating capability	✓	✓	✓	✓
Mesh-free simulation	✓	✓	✓	✓
Straightforward network training	✓	✗	✗	✗
Training without labeled data	✗	✓	✗	✗
Physics-informed loss function	✗	✓	✓	✓
Continuous solution	✗	✓	✓	✓
Spatiotemporal interpolation	✗	✓	✓	✓
Physics encoding	✗	✗	✓	✗
Efficient operator learning	✗	✗	✗	✓
Continuous-depth models	✗	✗	✓	✗
Spatiotemporal extrapolation	✗	✗	✓	✓
Solution transferability	✗	✗	✓	✓
Efficient real-time predictions	✗	✗	✗	✓

The combination of scientific computing and deep learning approaches also surpasses traditional computational mechanics solvers in a number of prevalent scenarios in practical engineering. For example, a sparse dataset obtained experimentally for a complex (i.e., hard-to-acquire data) phenomenon cannot be simply integrated with traditional solvers. By using DL, the following tasks can be performed: (i) PgNN-based models can be applied to the sparse data to extract latent interdependencies and conduct spatiotemporal downscaling or upscaling (i.e., interpolated data); (ii) PiNN-based models can be applied to the interpolated data to deduce governing equations and potentially unknown boundary or initial conditions of the phenomena (i.e., strong mathematical form); (iii) PeNN-based models can be used to combine the interpolated data and the strong mathematical form to conduct extrapolation exploration; and (iv) NO-based models can be applied to make real-time predictions of the complex dynamics. Therefore, the combination of DL-based methods and traditional scientific computing methods provides scientists with a cost-effective toolbox to explore problems across different scales that were deemed computationally far-fetched.

To this end, several other breakthroughs in DL are required to enable the use of PgNNs, PiNNs, PeNNs, and NOs in large-scale three-dimensional (or multidimensional) problems. For instance, the training of complex DL models (e.g., PiNNs, PeNNs, and NOs) should be accelerated using different parallelization paradigms, and the trained models should generalize well to unseen conditions, geometries, or material properties. Table 10 compares the main characteristics of PgNNs, PiNNs, PeNNs, and NOs. PgNN-based models suffer mainly from their statistical training process, for which they require large datasets. They map carefully curated training datasets only based on correlations in statistical variations, and hence, their predictions are naturally physics agnostic. PiNN-based models suffer mainly from the presence of competing loss terms that can destabilize the training process. PiNN is also a solution learning algorithm with limited generalizability due to its inability to learn the physical operation of a specific phenomenon. Models based on PeNNs and NOs, on the other hand, may experience low convergence rates and require a large volume of paired, structured datasets, leading to highly expensive training.

Considering the effectiveness of this new challenge of combining scientific computing and DL, future studies can be divided into three distinct categories: (i) *Improving algorithms*: Developing advanced variants of PgNNs, PiNNs, PeNNs, and NOs that offer simpler implementation with enhanced convergence rate; faster training in multidimensional and multiphysics problems; higher accuracy and generalization to unseen conditions while using sparse training datasets, more robust to be used in real time forecasting; better adaptability to multi-spatiotemporal resolutions, more flexibility to encode various types of governing equations (e.g., all PDE types, closure laws, data-driven laws, etc.), and provide a

closer tie with a plethora of traditional solvers. (ii) *Considering causalities*: Developing a causal training algorithm (e.g., causal Q-learning [351]) that restores physical causality during the training of PgNN, PiNN, and PeNN models by re-weighting the governing equations (e.g., PDEs) residual loss at each iteration. This line of research will allow for the development of causality-conforming variants of the PgNN, PiNN, and PeNN algorithms that can bring new opportunities for the application of these algorithms to a wider variety of complex scenarios across diverse domains. (iii) *Expanding applications*: Leveraging the potentials of PgNNs, PiNNs, PeNNs, and NOs in problems with complex anisotropic materials (e.g., flow in highly heterogeneous porous media, metal and non-metal particulate composites); problems with multiscale multiphysics phenomena (e.g., magnetorheological fluids, particle-laden fluids, dry powder dynamics, reactive transport, unsaturated soil dynamics); problems with multiresolution objectives and extensive spatiotemporal downscaling or upscaling (e.g., global and regional climate modeling, geosystem reservoir modeling); and structural health monitoring (e.g., crack identification and propagation, hydrogen pipeline leakage, CO_2 plume detection). and (iv) *Coupling solvers to form hybrid models*: Coupling PgNNs, PiNNs, and PeNNs as well as NOs with open-source computational mechanics packages such as OPENIFEM, OPENFOAM, PALABOS, LAMMPS, LIGGGHTS, and MOOSE. Integrating NNs with existing simulation tools and workflows seamlessly is a nontrivial task. This line of research will allow for faster surrogate modeling, and hence faster development of next-generation solvers. It also accelerates community and industry adoption of the combined scientific-DL computational paradigm.

Acknowledgment

S. A. F. would like to acknowledge supports from the Department of Energy Biological and Environmental Research (BER) (Award No. DE-SC0023044), National Science Foundation Partnership for Research and Education in Materials (PREM) (Award No. DMR-2122041), American Chemical Society Petroleum Research Fund (Award No. 66452-DNI9), and Texas State University Multidisciplinary Internal Research Grant (MIRG) (Award No. 9000003028). C. F. would like to acknowledge supports from FEDER funds through the COMPETE 2020 Programme and National Funds through FCT (Portuguese Foundation for Science and Technology) under the projects UID- B/05256/2020, UID-P/05256/2020 and MIT-EXPL/TDI/0038/2019-APROVA-Deep learning for particle-laden viscoelastic flow modelling (POCI-01-0145-FEDER-016665) under the MIT Portugal program.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The authors attest that all data for this study are included in the paper.

References

[1] Vinuesa, R., and Brunton, S. L., 2022, "Enhancing Computational Fluid Dynamics With Machine Learning," Nat. Comput. Sci., **2**(6), pp. 358–366.
[2] Mianroodi, J. R., Siboni, N. H., and Raabe, D., 2021, "Teaching Solid Mechanics to Artificial Intelligence—A Fast Solver for Heterogeneous Materials," npj Comput. Mater., **7**(1), p. 99.
[3] Kim, Y., Yang, C., Park, K., Gu, G. X., and Seunghwa, R., 2021, "Deep Learning Framework for Material Design Space Exploration Using Active Transfer Learning and Data Augmentation," npj Comput. Mater., **7**(1), p. 140.
[4] Liu, D., and Wang, Y., 2019, "Multi-fidelity Physics-Constrained Neural Network and its Application in Materials Modeling," ASME J. Mech. Des., **141**(12), p. 121403.
[5] Dino, H. I., Zeebaree, S. R., Salih, A. A., Zebari, R. R., Ageed, Z. S., Shukur, H. M., Haji, L. M., and Hasan, S. S., 2020, "Impact of Process Execution and Physical Memory-Spaces on Os Performance," Technol. Rep. Kansai Univ., **62**(5), pp. 2391–2401.
[6] Im, S., Lee, J., and Cho, M., 2021, "Surrogate Modeling of Elasto-Plastic Problems Via Long Short-Term Memory Neural Networks and Proper Orthogonal Decomposition," Comput. Methods Appl. Mech. Eng., **385**, p. 114030.
[7] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L., 2021, "Physics-Informed Machine Learning," Nat. Rev. Phys., **3**(6), pp. 422–440.
[8] Seyed-Ahmadi, A., and Wachs, A., 2022, "Physics-Inspired Architecture for Neural Network Modeling of Forces and Torques in Particle-Laden Flows," Comput. Fluids, **238**, p. 105379.
[9] Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., and Tebbutt, W., 2019, "A Differentiable Programming System to Bridge Machine Learning and Scientific Computing," arXiv preprint arXiv:1907.07587.
[10] Brunton, S. L., Noack, B. R., and Koumoutsakos, P., 2020, "Machine Learning for Fluid Mechanics," Annu. Rev. Fluid Mech., **52**, pp. 477–508.
[11] Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E., 2022, "Physics-Informed Neural Networks (PINNs) for Fluid Mechanics: A Review," Acta Mech. Sin., **37**(12), pp. 1727–1738.
[12] Kutz, J. N., 2017, "Deep Learning in Fluid Dynamics," J. Fluid Mech., **814**, pp. 1–4.
[13] Ayli, Ece, Kocak, Eyup, and Turkoglu, Hasmet, 2023, "Machine Learning Based Developing Flow Control Technique Over Circular Cylinders," ASME J. Comput. Inf. Sci. Eng., **23**(2), p. 021015.
[14] Shi, Z., Tsymbalov, E., Dao, M., Suresh, S., Shapeev, A., and Li, J., 2019, "Deep Elastic Strain Engineering of Bandgap Through Machine Learning," Proc. Natl. Acad. Sci. U. S. A., **116**(10), pp. 4117–4122.
[15] Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R., 2021, "A Physics-Informed Deep Learning Framework for Inversion and Surrogate Modeling in Solid Mechanics," Comput. Methods Appl. Mech. Eng., **379**, p. 113741.
[16] Chen, H., Nie, Z., Xu, Q., Fei, J., Yang, K., Li, Y., Lin, H., Fan, W., and Liu, X., 2023, "Intelligent Detection and Classification of Surface Defects on Cold-Rolled Galvanized Steel Strips Using a Data-Driven Faulty Model With Attention Mechanism," ASME J. Comput. Inf. Sci. Eng., **23**(4), p. 041001.
[17] Pilania, G., Wang, C., Jiang, X., Rajasekaran, S., and Ramprasad, R., 2013, "Accelerating Materials Property Predictions Using Machine Learning," Sci. Rep., **3**(1), p. 2810.
[18] Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O., and Walsh, A., 2018, "Machine Learning for Molecular and Materials Science," Nature, **559**(7715), pp. 547–555.
[19] Brunton, S. L., and Kutz, J. N., 2019, "Methods for Data-Driven Multiscale Model Discovery for Materials," J. Phys. Mater., **2**(4), p. 044002.
[20] Michopoulos, J. G., Hermanson, J. C., Iliopoulos, A., Lambrakos, S. G., and Furukawa, T., 2011, "Data-Driven Design Optimization for Composite Material Characterization," ASME J. Comput. Inf. Sci. Eng., **11**(2), p. 021009.
[21] Bedolla, E., Padierna, L. C., and Castaneda-Priego, R., 2020, "Machine Learning for Condensed Matter Physics," J. Phys.: Condens. Matter, **33**(5), p. 053001.
[22] Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S., 2021, "Machine Learning–Accelerated Computational Fluid Dynamics," Proc. Natl. Acad. Sci. U. S. A., **118**(21), p. e2101784118.
[23] Gandomi, A. H., Mignolet, M., Soize, C., and Wang, Y., 2023, "Machine Intelligence for Engineering Under Uncertainties," ASME J. Comput. Inf. Sci. Eng., **23**(1), p. 010201.
[24] Tran, D., Hoffman, M. D., Saurous, R. A., Brevdo, E., Murphy, K., and Blei, D. M., 2017, "Deep Probabilistic Programming," arXiv preprint arXiv:1701.03757.
[25] Jin, K. H., McCann, M. T., Froustey, E., and Unser, M., 2017, "Deep Convolutional Neural Network for Inverse Problems in Imaging," IEEE Trans. Image. Process., **26**(9), pp. 4509–4522.
[26] Warey, A., Raul, V., Kaushik, S., Han, T., and Chakravarty, R., 2023, "Generative Inverse Design of Aerodynamic Shapes Using Conditional Invertible Neural Networks," ASME J. Comput. Inf. Sci. Eng., **23**(3), p. 031006.
[27] Lai, Z., Chen, Q., and Huang, L., 2022, "Machine-Learning-Enabled Discrete Element Method: Contact Detection and Resolution of Irregular-Shaped Particles," Int. J. Numer. Anal. Methods Geomech., **46**(1), pp. 113–140.
[28] Faroughi, S. A., Roriz, A. I., and Fernandes, C., 2022, "A Meta-Model to Predict the Drag Coefficient of a Particle Translating in Viscoelastic Fluids: A Machine Learning Approach," Polymers, **14**(3), p. 430.
[29] Taylor, B., 1717, "Methodus Incrementorum Directa & Inversa. Auctore Brook Taylor, LL. D. & Regiae Societatis Secretario," typis Pearsonianis: prostant apud Gul. Innys ad Insignia Principis in ….
[30] Alder, B. J., and Wainwright, T. E., 1957, "Phase Transition for a Hard Sphere System," J. Chem. Phys., **27**(5), pp. 1208–1209.
[31] Clough, R. W., 1960, "The Finite Element Method in Plane Stress Analysis," Proceedings of 2nd ASCE Conference on Electronic Computation, Pittsburgh, PA, Sept. 8–9.
[32] Smagorinsky, J., 1963, "General Circulation Experiments With the Primitive Equations: I. The Basic Experiment," Mon. Weather Rev., **91**(3), pp. 99–164.
[33] Cundall, P. A., and Strack, O., 1979, "A Discrete Numerical Model for Granular Assemblies," Géotechnique, **29**(1), pp. 47–65.
[34] McDonald, P. W., 1971, *The Computation of Transonic Flow Through Two-Dimensional Gas Turbine Cascades*, Vol. 79825, American Society of Mechanical Engineers, p. V001T01A089.

[35] Peskin, C. S., 1972, "Flow Patterns Around Heart Valves: A Numerical Method," J. Comput. Phys., **10**(2), pp. 252–271.
[36] Lucy, L. B., 1977, "A Numerical Approach to the Testing of the Fission Hypothesis," Astron. J., **82**, pp. 1013–1024.
[37] D'Humieres, D., Lallemand, P., and Frisch, U., 1986, "Lattice Gas Models for 3D Hydrodynamics," EPL (Europhys. Lett.), **2**(4), p. 291.
[38] Bassi, F., and Rebay, S., 1997, "A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations," J. Comput. Phys., **131**(2), pp. 267–279.
[39] Ivakhnenko, A. G., and Lapa, V. G., 1967, *Cybernetics and Forecasting Techniques*, Vol. 8, American Elsevier Publishing Company.
[40] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986, "Learning Representations by Back-Propagating Errors," Nature, **323**(6088), pp. 533–536.
[41] Andersen, K., Cook, G. E., Karsai, G., and Ramaswamy, K., 1990, "Artificial Neural Networks Applied to ARC Welding Process Modeling and Control," IEEE Trans. Ind. Appl., **26**(5), pp. 824–830.
[42] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., 1998, "Gradient-Based Learning Applied to Document Recognition," Proc. IEEE, **86**(11), pp. 2278–2324.
[43] Goodfellow, I. J., Jean, P., Mehdi, M., Bing, X., David, W., Sherjil, O., and Courville Aaron, C., 2014, "Generative Adversarial Nets," Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, Canada, Dec. 8–13, Vol. 2, pp. 2672–2680.
[44] Raissi, M., Perdikaris, P., and Karniadakis, G. E., 2017, "Physics Informed Deep Learning (Part I): Data-Driven Solutions of Nonlinear Partial Differential Equations," arXiv preprint arXiv:1711.10561.
[45] Lu, L., Jin, P., and Karniadakis, G. E., 2019, "Deeponet: Learning Nonlinear Operators for Identifying Differential Equations Based on the Universal Approximation Theorem of Operators," arXiv preprint arXiv:1910.03193.
[46] Rao, C., Sun, H., and Liu, Y., "Hard Encoding of Physics for Learning Spatiotemporal Dynamics," arXiv preprint arXiv:2105.00557.
[47] Zhao, Y., Jiang, C., Vega, M. A., Todd, M. D., and Hu, Z., 2023, "Surrogate Modeling of Nonlinear Dynamic Systems: A Comparative Study," ASME J. Comput. Inf. Sci. Eng., **23**(1), p. 011001.
[48] Lienen, M., and Gunnemann, S., 2022, "Learning the Dynamics of Physical Systems From Sparse Observations With Finite Element Networks," The Tenth International Conference on Learning Representations, Virtual, Apr. 25–29, OpenReview.
[49] Hasson, U., Nastase, S. A., and Goldstein, A., 2020, "Direct Fit to Nature: An Evolutionary Perspective on Biological and Artificial Neural Networks," Neuron, **105**(3), pp. 416–434.
[50] Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A., 2021, "Physics-Informed Neural Operator for Learning Partial Differential Equations," arXiv preprint arXiv:2111.03794.
[51] Raissi, M., Perdikaris, P., and Karniadakis, G. E., 2019, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," J. Comput. Phys., **378**, pp. 686–707.
[52] Nabian, M. A., and Meidani, H., 2020, "Adaptive Physics-Informed Neural Networks for Markov-Chain Monte Carlo," arXiv preprint arXiv:2008.01604.
[53] Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F., 2022, "Scientific Machine Learning Through Physics-Informed Neural Networks: Where We Are and What'a Next," J. Sci. Comput., **92**(3), p. 88.
[54] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M., 2015, "Automatic Differentiation in Machine Learning: A Survey," J. Mach. Learn. Res., **18**, pp. 1–43.
[55] Rao, C., Hao, S., and Yang, L., 2020, "Physics-Informed Deep Learning for Incompressible Laminar Flows," Theor. Appl. Mech. Lett., **10**(3), pp. 207–212.
[56] Faroughi, S. A, Soltanmohammadi, R., Datta, P., Mahjour, S. K., and Faroughi, S., 2023, "Physics-Informed Neural Networks With Periodic Activation Functions for Solute Transport in Heterogeneous Porous Media," Mathematics, **12**(1), p. 63.
[57] McClenny, L., and Braga-Neto, U., 2020, "Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism," arXiv preprint arXiv:2009.04544.
[58] Yadav, G. K., Natarajan, S., and Srinivasan, B., 2022, "Distributed Pinn for Linear Elasticity—a Unified Approach for Smooth, Singular, Compressible and Incompressible Media," Int. J. Comput. Methods, p. 2142008.
[59] Bauer, P., Dueben, P. D., Hoefler, T., Quintino, T., Schulthess, T. C., and Wedi, N. P., 2021, "The Digital Revolution of Earth-System Science," Nat. Comput. Sci., **1**(2), pp. 104–113.
[60] Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K., 2018, "Neural Ordinary Differential Equations," Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 6572–6583.
[61] Chung, H., Lee, S. J., and Park, J. G., 2016, "Deep Neural Network Using Trainable Activation Functions," 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, July 24–29, IEEE, pp. 348–352.
[62] Mattheakis, M., Protopapas, P., Sondak, D., Di, G. M., and Kaxiras, E., 2019, "Physical Symmetries Embedded in Neural Networks," arXiv preprint arXiv:1904.08991.
[63] Lu, L., Pengzhan, J., Pang, G., Zhang, Z., and Karniadakis, G. E., 2021, "Learning Nonlinear Operators Via Deeponet Based on the Universal Approximation Theorem of Operators," Nat. Mach. Intell., **3**(3), pp. 218–229.
[64] Goswami, S., Bora, A., Yu, Y., and Karniadakis, G. E., 2022, "Physics-Informed Neural Operators," arXiv preprint arXiv:2207.05748.
[65] Wang, S., Wang, H., and Perdikaris, P., 2021, "Learning the Solution Operator of Parametric Partial Differential Equations With Physics-Informed Deeponets," Sci. Adv., **7**(40), p. eabi8605.
[66] LeCun, Y., Bengio, Y., and Hinton, G., 2015, "Deep Learning," Nature, **521**(7553), pp. 436–444.
[67] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A., 2018, "Generative Adversarial Networks: An Overview," IEEE Signal Process. Mag., **35**(1), pp. 53–65.
[68] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G., 2008, "The Graph Neural Network Model," IEEE Trans. Neural Netw., **20**(1), pp. 61–80.
[69] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2020, "Generative Adversarial Networks," Commun. ACM, **63**(11), pp. 139–144.
[70] Rasamoelina, A. D., Adjailia, F., and Sinčák, P., 2020, "A Review of Activation Function for Artificial Neural Network," 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, Jan. 23–25, IEEE, pp. 281–286.
[71] He, C., Ma, M., and Wang, P., 2020, "Extract Interpretability-Accuracy Balanced Rules From Artificial Neural Networks: A Review," Neurocomputing, **387**, pp. 346–358.
[72] Li, M., Zhang, T., Chen, Y., and Smola, A. J., 2014, "Efficient Mini-Batch Training for Stochastic Optimization," Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, Aug. 24–27, pp. 661–670.
[73] Wessels, H., Weißenfels, C., and Wriggers, P., 2020, "The Neural Particle Method–an Updated Lagrangian Physics Informed Neural Network for Computational Fluid Dynamics," Comput. Methods Appl. Mech. Eng., **368**, p. 113127.
[74] Huang, K., Krugener, M., Brown, A., Menhorn, F., Bungartz, H. J., and Hartmann, D., 2021, "Machine Learning-Based Optimal Mesh Generation in Computational Fluid Dynamics," arXiv preprint: arXiv:2102.12923.
[75] Kumar, S., and Kochmann, D. M., 2022, "What Machine Learning Can Do for Computational Solid Mechanics," *Current Trends and Open Problems in Computational Mechanics*, F. Aldakheel, B. Hudobivnik, M. Soleimani, H. Wessels, C. Weißenfels, and M. Marino, eds., Springer, Cham, pp. 275–285.
[76] Guo, K., Yang, Z., Yu, C-H., and Buehler, M. J., 2021, "Artificial Intelligence and Machine Learning in Design of Mechanical Materials," Mater. Horiz., **8**(4), pp. 1153–1172.
[77] Zhang, Z., Wang, Y., Jimack, P. K., and Wang, H., 2020, "MeshingNet: A New Mesh Generation Method Based on Deep Learning," International Conference on Computational Science, Springer International Publishing, Cham, pp. 186–198.
[78] Wu, T., Liu, X., An, W., Huang, Z., and Lyu, H., 2022, "A Mesh Optimization Method Using Machine Learning Technique and Variational Mesh Adaptation," Chin. J. Aeronaut., **35**(3), pp. 27–41.
[79] Mendizabal, A., Márquez-Neila, P., and Cotin, S., 2020, "Simulation of Hyperelastic Materials in Real-Time Using Deep Learning," Med. Image Anal., **59**, p. 101569.
[80] Lu, L., Gao, X., Dietiker, J-F., Shahnam, M., and Rogers, W. A., 2021, "Machine Learning Accelerated Discrete Element Modeling of Granular Flows," Chem. Eng. Sci., **245**, p. 116832.
[81] Li, Z., Meidani, K., Yadav, P., and Farimani, A. B., 2021, "Graph Neural Networks Accelerated Molecular Dynamics," J. Chem. Phys., **156**(14).
[82] Menke, H. P., Maes, J., and Geiger, S., 2021, "Upscaling the Porosity-Permeability Relationship of a Microporous Carbonate for Darcy-Scale Flow With Machine Learning," Sci. Rep., **11**(1), pp. 1–10.
[83] Cheng, S., Chen, J., Anastasiou, C., Angeli, P., Matar, O. K., Guo, Y., Pain, C. C., and Arcucci, R., 2023, "Generalised Latent Assimilation in Heterogeneous Reduced Spaces With Machine Learning Surrogate Models," J. Sci. Comput., **94**(1), p. 11.
[84] Zawawi, M. H., Saleha, A., Salwa, A., Hassan, N. H., Zahari, N. M., Ramli, M. Z., and Muda, Z. C., 2018, "A Review: Fundamentals of Computational Fluid Dynamics (CFD)," *AIP Conference Proceedings*, M. M.Al-Bakri Abdullah, S. Z. Bin Abd. Rahim, M. N. Bin Mat Saad, M. Fathullah bin Ghazli, R. Ahmad, M. Faheem Bin Mohd Tahir and L. B. Jamaludin, eds., Vol. 2030, AIP Publishing LLC, p. 020252.
[85] He, L., and Tafti, D. K., 2019, "A Supervised Machine Learning Approach for Predicting Variable Drag Forces on Spherical Particles in Suspension," Powder Technol., **345**, pp. 379–389.
[86] Zhu, L-T., Tang, J-X., and Luo, Z-H., 2020, "Machine Learning to Assist Filtered Two-Fluid Model Development for Dense Gas–Particle Flows," AIChE J., **66**(6), p. e16973.
[87] Roriz, A. I., Faroughi, S. A., McKinley, G. H., and Fernandes, C., 2021, "ML Driven Models to Predict the Drag Coefficient of a Sphere Translating in Shear-Thinning Viscoelastic Fluids".
[88] Loiro, C., Fernandes, C., McKinley, G. H., and Faroughi, S. A., 2021, "Digital-Twin for Particle-Laden Viscoelastic Fluids: Ml-Based Models to Predict the Drag Coefficient of Random Arrays of Spheres."
[89] Webb, M. A., Jackson, N. E., Gil, P. S., and de Pablo, J. J., 2020, "Targeted Sequence Design Within the Coarse-Ggrained Polymer Genome," Sci. Adv., **6**(43), p. eabc6216.
[90] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., 2014, "Dropout: A Simple Way to Prevent Neural Networks From Overfitting," J. Mach. Learn. Res., **15**(1), pp. 1929–1958.

[91] Bejani, M. M., and Ghatee, M., 2021, "A Systematic Review on Overfitting Control in Shallow and Deep Neural Networks," Artif. Intell. Rev., **54**(8), pp. 6391–6438.
[92] Ying, X., 2019, "An Overview of Overfitting and Its Solutions," *Journal of Physics: Conference Series*, M. Land, ed., Vol. 1168, IOP Publishing, p. 022022.
[93] Cati, Y., aus der Wiesche, S., and Düzgün, M., 2022, "Numerical Model of the Railway Brake Disk for the Temperature and Axial Thermal Stress Analyses," ASME J. Therm. Sci. Eng. Appl., **14**(10), p. 101014.
[94] Chen, X., Liu, J., Pang, Y., Chen, J., Chi, L., and Gong, C., 2020, "Developing a New Mesh Quality Evaluation Method Based on Convolutional Neural Network," Eng. Applic. Comput. Fluid Mech., **14**(1), pp. 391–400.
[95] Maddu, S., Sturm, D., Cheeseman, B. L., Müller, C. L., and Sbalzarini, I. F., 2021, "Stencil-Net: Data-Driven Solution-Adaptive Discretization of Partial Differential Equations," arXiv preprint arXiv:2101.06182.
[96] Bar-Sinai, Y., Hoyer, S., Hickey, J., and Brenner, M. P., 2019, "Learning Data-Driven Discretizations for Partial Differential Equations," Proc. Natl. Acad. Sci. U. S. A., **116**(31), pp. 15344–15349.
[97] Bernardin, F., Bossy, M., Chauvin, C., Jabir, J-F., and Rousseau, A., 2010, "Stochastic Lagrangian Method for Downscaling Problems in Computational Fluid Dynamics," ESAIM: Math. Model. Numer. Anal., **44**(5), pp. 885–920.
[98] Wei, X., Dong, C., Chen, Z., Xiao, K., and Li, X., 2016, "The Effect of Hydrogen on the Evolution of Intergranular Cracking: A Cross-scale Study Using First-Principles and Cohesive Finite Element Methods," RSC Adv., **6**(33), pp. 27282–27292.
[99] Cockburn, B., Shu, C. W., Johnson, C., Tadmor, E., and Shu, C. W., 1998, "Essentially Non-oscillatory and Weighted Essentially Non-oscillatory Schemes for Hyperbolic Conservation Laws," *Advanced Numerical Approximation of Nonlinear Hyberbolic Equations*, Springer, Berlin/Heidelberg, pp. 325–432.
[100] Zhang, Z., Jimack, P. K., and Wang, H., 2021, "MeshingNet3D: Efficient Generation of Adapted Tetrahedral Meshes for Computational Mechanics," Adv. Eng. Softw., **157**, p. 103021.
[101] Triantafyllidis, D. G., and Labridis, D. P., 2002, "A Finite-Element Mesh Generator Based on Growing Neural Networks," IEEE Transac. Neural Networks, **13**(6), pp. 1482–1496.
[102] Srasuay, K., Chumthong, A., and Ruangsinchaiwanich, S., 2010, "Mesh Generation of FEM by Ann on Iron—core Transformer," 2010 International Conference on Electrical Machines and Systems, Incheon, South Korea, Oct. 10–13, IEEE, pp. 1885–1890.
[103] Xu, H., Nie, Z., Xu, Q., Li, Y., Xie, F., and Liu, X-J., 2023, "Supermeshing: Boosting the Mesh Density of Stress Field in Plane-Strain Problems Using Deep Learning Method," ASME J. Comput. Inf. Sci. Eng., **23**(3), p. 034501.
[104] Lee, M. J., and Chen, J. T., 1993, "Fluid Property Predictions With the Aid of Neural Networks," Ind. Eng. Chem. Res., **32**(5), pp. 995–997.
[105] Yang, C., Yang, X., and Xiao, X., 2016, "Data-Driven Projection Method in Fluid Simulation," Comput. Anim. Virtual Worlds, **27**(3–4), pp. 415–424.
[106] Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K., 2016, "Accelerating Eulerian Fluid Simulation with Convolutional Networks," International Conference on Machine Learning, PMLR, pp. 3424–3433.
[107] Jacobs, D. A. H., 1980, "Preconditioned Conjugate Gradient Methods for Solving Systems of Algebraic Equations," Central Electricity Research Laboratories, Technical Report RD/L/N193/80.
[108] Chen, J., Viquerat, J., and Hachem, E., 2019, "U-Net Architectures for Fast Prediction of Incompressible Laminar Flows," arXiv preprint arXiv:1910.13532.
[109] Deng, Z., He, C., Liu, Y., and Kim, K., 2019, "Super-Resolution Reconstruction of Turbulent Velocity Fields Using a Generative Adversarial Network-Based Artificial Intelligence Framework," Phys. Fluids, **31**(12), p. 125111.
[110] Ling, J., Kurzawski, A., and Templeton, J., 2016, "Reynolds Averaged Turbulence Modelling Using Deep Neural Networks with Embedded Invariance," J. Fluid Mech., **807**, pp. 155–166.
[111] Lévy-Leblond, J., 1971, "Galilei Group and Galilean Invariance," *Group Theory and its Applications*, E. M. Loebl, ed., Elsevier, Amsterdam, pp. 221–299.
[112] Maulik, R., San, O., Rasheed, A., and Vedula, P., 2019, "Subgrid Modelling for Two-Dimensional Turbulence Using Neural Networks," J. Fluid Mech., **858**, pp. 122–144.
[113] Kraichnan, R. H., 1967, "Inertial Ranges in Two-Dimensional Turbulence," Phys. Fluids, **10**(7), pp. 1417–1423.
[114] Kim, J., and Lee, C., 2020, "Prediction of Turbulent Heat Transfer Using Convolutional Neural Networks," J. Fluid Mech., **882**, p. A18.
[115] Kingma, D. P., and Ba. Adam, J., 2014, "A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980.
[116] Hoang, N-D., 2020, "Image Processing-based Spall Object Detection Using Gabor Filter, Texture Analysis, and Adaptive Moment Estimation (ADAM) Optimized Logistic Regression Models," Adv. Civil Eng., **2020**, pp. 1–16.
[117] Priyadarshini, I., and Cotton, C., 2021, "A Novel LSTM–CNN-Grid Search-Based Deep Neural Network for Sentiment Analysis," J. Supercomput., **77**(12), pp. 13911–13932.
[118] Sun, Y., Ding, S., Zhang, Z., and Jia, W., 2021, "An Improved Grid Search Algorithm to Optimize SVR for Prediction," Soft Comput., **25**(7), pp. 5633–5644.
[119] Yousif, M., Yu, L., and Lim, H., 2022, "Physics-Guided Deep Learning for Generating Turbulent Inflow Conditions," J. Fluid Mech., **936**, p. A21.
[120] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z., 2016, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-pixel Convolutional Neural Network," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, June 17–24, pp. 1874–1883.
[121] Talab, M. A., Awang, S., and Najim, S. M., 2019, "Super-Low Resolution Face Recognition Using Integrated Efficient Sub-Pixel Convolutional Neural Network (ESPCN) and Convolutional Neural Network (CNN)," 2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), Shah Alam, Selangor, Malaysia, June 29, IEEE, pp. 331–335.
[122] Huang, Z., Xu, W., and Yu, K., 2015, "Bidirectional LSTM-CRF Models for Sequence Tagging," arXiv preprint arXiv:1508.01991.
[123] Sherstinsky, A., 2020, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," Physica D: Nonlinear Phenom., **404**, p. 132306.
[124] Kou, J., and Zhang, W., 2021, "Data-Driven Modeling for Unsteady Aerodynamics and Aeroelasticity," Prog. Aerosp. Sci., **125**, p. 100725.
[125] Wang, Z., Gong, K., Fan, W., Li, C., and Qian, W., 2022, "Prediction of Swirling Flow Field in Combustor Based on Deep Learning," Acta Astronaut., **201**, pp. 302–316.
[126] Chowdhary, K., Hoang, C., Lee, K., Ray, J., Weirs, V. G., and Carnes, B., 2022, "Calibrating Hypersonic Turbulence Flow Models With the HIFiRE-1 Experiment Using Data-Driven Machine-Learned Models," Comput. Methods Appl. Mech. Eng., **401**, p. 115396.
[127] Bond, B. N., and Daniel, L., 2008, "Guaranteed Stable Projection-Based Model Reduction for Indefinite and Unstable Linear Systems," 2008 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, Nov. 10–13, IEEE, pp. 728–735.
[128] Beli, D., Mencik, J-M., Silva, P. B., and Arruda, J., 2018, "A Projection-Based Model Reduction Strategy for the Wave and Vibration Analysis of Rotating Periodic Structures," Comput. Mech., **62**(6), pp. 1511–1528.
[129] Siddiqui, M. F., De Troyer, T., Decuyper, J., Csurcsia, P. Z., Schoukens, J., and Runacres, M. C., 2022, "A Data-Driven Nonlinear State-Space Model of the Unsteady Lift Force on a Pitching Wing," J. Fluids Struct., **114**, p. 103706.
[130] Wang, X., Kou, J., and Zhang, W., 2022, "Unsteady Aerodynamic Prediction for Iced Airfoil Based on Multi-task Learning," Phys. Fluids, **34**(8), p. 087117.
[131] Stevens, B., and Colonius, T., 2020, "Enhancement of Shock-Capturing Methods Via Machine Learning," Theor. Comput. Fluid Dyn., **34**(4), pp. 483–496.
[132] Pawar, N., and Faroughi, S. A., 2022, "Complex Fluids Latent Space Exploration Towards Accelerated Predictive Modeling," Bulletin of the American Physical Society.
[133] Fernandes, C., Faroughi, S. A., Ferrás, L. L., and Afonso, A. M., 2022, "Advanced Polymer Simulation and Processing," Polymers, **14**(12), p. 2480
[134] Pawar, N., Mahjour, S. K., Kalantari, N. K., and Faroughi, S., 2022, "Spatiotemporal Down-scaling for Multiphase Flow in Porous Media Using Implicit Hypernetwork Neural Representation," AGU Fall Meeting Abstracts, Chicago, IL, Dec. 12–16, Vol. 2022, pp. H45M–1555.
[135] Faroughi, S. A., Fernandes, C., Nóbrega, J. M., and McKinley, G. H., 2020, "A Closure Model for the Drag Coefficient of a Sphere Translating in a Viscoelastic Fluid," J. Non-Newtonian Fluid Mech., **277**, p. 104218.
[136] Fernandes, C., Faroughi, S. A., Carneiro, O. S., Miguel Nóbrega, J., and McKinley, G. H., 2019, "Fully-Resolved Simulations of Particle-Laden Viscoelastic Fluids Using an Immersed Boundary Method," J. Non-Newtonian Fluid Mech., **266**, pp. 80–94.
[137] Lin, W., Wu, Z., Lin, L., Wen, A., and Li, J., 2017, "An Ensemble Random Forest Algorithm for Insurance Big Data Analysis," IEEE Access, **5**, pp. 16568–16575.
[138] Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., et al. 2015, "Xgboost: Extreme Gradient Boosting," R Package Version 0.4-2, **1**(4), pp. 1–4.
[139] Lennon, K. R., McKinley, G. H., and Swan, J. W., 2022, "Scientific Machine Learning for Modeling and Simulating Complex Fluids," Proc. Natl. Acad. Sci. U.S.A., **120**(27), p. e2304669120.
[140] Cai, Z., Chen, J., and Liu, M., 2022, "Least-Squares ReLU Neural Network (LSNN) Method for Scalar Nonlinear Hyperbolic Conservation Law," Appl. Numer. Math., **174**, pp. 163–176.
[141] El Haber, G., Viquerat, J., Larcher, A., Ryckelynck, D., Alves, J., Patil, A., and Hachem, E., 2022, "Deep Learning Model to Assist Multiphysics Conjugate Problems," Phys. Fluids, **34**(1), p. 015131.
[142] Lara, F. M., and Ferrer, E., 2022, "Accelerating High Order Discontinuous Galerkin Solvers Using Neural Networks: 1D Burgers' Equation," Comput. Fluids, **235**, p. 105274.
[143] List, B., Chen, L., and Thuerey, N., 2022, "Learned Turbulence Modelling With Differentiable Fluid Solvers: Physics-Based Loss Functions and Optimisation Horizons," J. Fluid Mech., **949**, p. A25.
[144] Beck, A. D., Flad, D. G., and Munz, C. D., 2018, "Deep Neural Networks for Data-Driven Turbulence Models," Proceedings of the APS Division of Fluid Dynamics Meeting Abstracts, p. G16.
[145] Sekar, V., Jiang, Q. H., Shu, C., and Khoo, B. C., 2019, "Fast Flow Field Prediction Over Airfoils Using Deep Learning Approach," Phys. Fluids, **31**(5), p. 057103.
[146] Cao, M., Wang, K. W., DeVries, L., Fujii, Y., Tobler, W. E., Pietron, G. M., Tibbles, T., and McCallum, J., 2004, "Steady State Hydraulic Valve Fluid Field Estimator Based on Non-Dimensional Artificial Neural Network (NDANN)," ASME J. Comput. Inf. Sci. Eng., **4**(3), pp. 257–270.
[147] Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., and Gupta, S. K., 2021, "Image-Based Surface Defect Detection Using Deep Learning: A Review," ASME J. Comput. Inf. Sci. Eng., **21**(4), p. 040801.

[148] Tadesse, Z., Patel, K., Chaudhary, S., and Nagpal, A., 2012, "Neural Networks for Prediction of Deflection in Composite Bridges," J. Constr. Steel Res., **68**(1), pp. 138–149.
[149] Güneyisi, E. M., D'Aniello, M., Landolfo, R., Mermerdaş, K., et al. 2014, "Prediction of the Flexural Overstrength Factor for Steel Beams Using Artificial Neural Network," Steel and Compos. Struct., **17**(3), pp. 215–236.
[150] Hung, T. V., Viet, V. Q., and Van, T. D., 2019, "A Deep Learning-Based Procedure for Estimation of Ultimate Load Carrying of Steel Trusses Using Advanced Analysis," J. Sci. Technol. Civil Eng. (STCE)-HUCE, **13**(3), pp. 113–123.
[151] Chen, G., Li, T., Chen, Q., Ren, S., Wang, C., and Li, S., 2019, "Application of Deep Learning Neural Network to Identify Collision Load Conditions Based on Permanent Plastic Deformation of Shell Structures," Comput. Mech., **64**(2), pp. 435–449.
[152] Hosseinpour, M., Sharifi, Y., and Sharifi, H., 2020, "Neural Network Application for Distortional Buckling Capacity Assessment of Castellated Steel Beams," *Structures*, Vol. 27, L. Gardner, ed., Elsevier, pp. 1174–1183.
[153] Trahair, N. S., and Bradford, M. A., 2017, *The Behaviour and Design of Steel Structures to AS 4100*, CRC Press, Boca Raton, FL.
[154] White, D. W., Surovek, A. E., Alemdar, B. N., Chang, C-J., Kim, Y. D., and Kuchenbecker, G. H., 2006, "Stability Analysis and Design of Steel Building Frames Using the 2005 AISC Specification," Steel Struct., **6**(2), pp. 71–91.
[155] European Committee for Standardization (ECS), 2005, "Design of Steel Structures Part 1–1: General Rules and Rules for Buildings," Wiley, Brussels, Belgium.
[156] White, D. A., Arrighi, W. J., Kudo, J., and Watts, S. E., 2019, "Multiscale Topology Optimization Using Neural Network Surrogate Models," Comput. Methods Appl. Mech. Eng., **346**, pp. 1118–1135.
[157] Zhang, Y., Li, H., Xiao, M., Gao, L., Chu, S., and Zhang, J., 2019, "Concurrent Topology Optimization for Cellular Structures With Nonuniform Microstructures Based on the Kriging Metamodel," Struct. Multidiscipl. Optim., **59**(4), pp. 1273–1299.
[158] Sigmund, O., and Maute, K., 2013, "Topology Optimization Approaches," Struct. Multidiscipl. Optim., **48**(6), pp. 1031–1055.
[159] Subedi, S. C., Verma, C. S., and Suresh, K., 2020, "A Review of Methods for the Geometric Post-Processing of Topology Optimized Models," ASME J. Comput. Inf. Sci. Eng., **20**(6), p. 060801.
[160] Abueidda, D. W., Koric, S., and Sobh, N. A., 2020, "Topology Optimization of 2D Structures With Nonlinearities Using Deep Learning," Comput. Struct., **237**, p. 106283.
[161] Yu, Y., Hur, T., Jung, J., and Jang, I. G., 2019, "Deep Learning for Determining a Near-Optimal Topological Design Without Any Iteration," Struct. Multidiscipl. Optim., **59**(3), pp. 787–799.
[162] Banga, S., Gehani, H., Bhilare, S., Patel, S., and Kara, L., 2018, "3D Topology Optimization using Convolutional Neural Networks," arXiv preprint arXiv:1808.07440.
[163] Li, B., Huang, C., Li, X., Zheng, S., and Hong, J., 2019, "Non-Iterative Structural Topology Optimization Using Deep Learning," Comput. Aided Design, **115**, pp. 172–180.
[164] Takano, N., and Alaghband, G., 2019, "Srgan: Training Dataset Matters," arXiv preprint arXiv:1903.09922.
[165] Nagano, Y., and Kikuta, Y., 2018, "Srgan for Super-Resolving Low-Resolution Food Images," Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management, Stockholm, Sweden, July 15, pp. 33–37.
[166] Messner, M. C., 2020, "Convolutional Neural Network Surrogate Models for the Mechanical Properties of Periodic Structures," ASME J. Mech. Des., **142**(2), p. 024503.
[167] Tcherniak, D., 2002, "Topology Optimization of Resonating Structures Using Simp Method," Int. J. Numer. Methods Eng., **54**(11), pp. 1605–1622.
[168] Lininger, A., Hinczewski, M., and Strangi, G., 2021, "General Inverse Design of Thin-Film Metamaterials With Convolutional Neural Networks," arXiv preprint arXiv:2104.01952.
[169] Löper, P., Stuckelberger, M., Niesen, B., Werner, J., Filipič, M., Moon, S., Yum, J-H., Topič, M., De Wolf, S., and Ballif, C., 2014, "Complex Refractive Index Spectra of CH3NH3PbI3 Perovskite Thin Films Determined by Spectroscopic Ellipsometry and Spectrophotometry," J. Phys. Chem. Lett., **6**(1), pp. 66–71.
[170] Kumar, S., Tan, S., Zheng, L., and Kochmann, D. M., 2020, "Inverse-Designed Spinodoid Metamaterials," npj Comput. Mater., **6**(1), pp. 1–10.
[171] Ni, B., and Gao, H., 2021, "A Deep Learning Approach to the Inverse Problem of Modulus Identification in Elasticity," MRS Bull., **46**(1), pp. 19–25.
[172] Smith, K. E., and Smith, A. O., 2020, "Conditional GAN for Timeseries Generation," arXiv preprint arXiv:2006.16477.
[173] Balaji, Y., Min, M. R., Bai, B., Chellappa, R., and Graf, H. P., 2019, "Conditional Gan With Discriminative Filter Generation for Text-to-Video Synthesis," *IJCAI*, O. Scherzer, IOP Publishing Ltd., Vol. 1, p. 2.
[174] Oberai, A. A., Gokhale, N. H., and Feijóo, G. R., 2003, "Solution of Inverse Problems in Elasticity Imaging Using the Adjoint Method," Inverse Problems, **19**(2), p. 297.
[175] Liang, L., Liu, M., Martin, C., and Sun, W., 2018, "A Deep Learning Approach to Estimate Stress Distribution: A Fast and Accurate Surrogate of Finite-Element Analysis," J. R. Soc. Interface, **15**(138), p. 20170844.
[176] Mozaffar, M., Bostanabad, R., Chen, W., Ehmann, K., Cao, J., and Bessa, M. A., 2019, "Deep Learning Predicts Path-Dependent Plasticity," PNAS, **116**, pp. 26414–26420.
[177] Chatterjee, A., 2000, "An Introduction to the Proper Orthogonal Decomposition," Curr. Sci. pp. 808–817.
[178] Liang, Y. C., Lee, H. P., Lim, S. P., Lin, W. Z., Lee, K. H., and Wu, C., 2002, "Proper Orthogonal Decomposition and Its Applications—Part I: Theory," J. Sound Vib., **252**(3), pp. 527–544.
[179] Long, X. Y., Zhao, S. K., Jiang, C., Li, W. P., and Liu, C. H., 2021, "Deep Learning-Based Planar Crack Damage Evaluation Using Convolutional Neural Networks," Eng. Fract. Mech., **246**, p. 107604.
[180] Zhu, S., Ohsaki, M., and Guo, X., 2021, "Prediction of Non-Linear Buckling Load of Imperfect Reticulated Shell Using Modified Consistent Imperfection and Machine Learning," Eng. Struct., **226**, p. 111374.
[181] Miller, B., and Ziemiański, L., 2020, "Optimization of Dynamic Behavior of Thin-Walled Laminated Cylindrical Shells by Genetic Algorithms and Deep Neural Networks Supported by Modal Shape Identification," Adv. Eng. Softw., **147**, p. 102830.
[182] Konovalenko, I., Maruschak, P., and Brevus, V., 2022, "Steel Surface Defect Detection Using an Ensemble of Deep Residual Neural Networks," ASME J. Comput. Inf. Sci. Eng., **22**(1), p. 014501.
[183] Nie, Z., Jiang, H., and Kara, L. B., 2020, "Stress Field Prediction in Cantilevered Structures Using Convolutional Neural Networks," ASME ASME J. Comput. Inf. Sci. Eng., **20**(1), p. 011002.
[184] Pizarro, P. N., and Massone, L. M., 2021, "Structural Design of Reinforced Concrete Buildings Based on Deep Neural Networks," Eng. Struct., **241**, p. 112377.
[185] Pathirage, C. S., Li, J., Li, L., Hao, H., Liu, W., and Ni, P., 2018, "Structural Damage Identification Based on Autoencoder Neural Networks and Deep Learning," Eng. Struct., **172**, pp. 13–28.
[186] Jiang, S., and Zhang, J., 2020, "Real-Time Crack Assessment Using Deep Neural Networks With Wall-Climbing Unmanned Aerial System," Comput. Aided Civil Infrastructure Eng., **35**(6), pp. 549–564.
[187] Perez-Ramirez, C. A., Amezquita-Sanchez, J. P., Valtierra-Rodriguez, M., Adeli, H., Dominguez-Gonzalez, A., and Romero-Troncoso, R. J., 2019, "Recurrent Neural Network Model With Bayesian Training and Mutual Information for Response Prediction of Large Buildings," Eng. Struct., **178**, pp. 603–615.
[188] Truong, T. T., Dinh-Cong, D., Lee, J., and Nguyen-Thoi, T., 2020, "An Effective Deep Feedforward Neural Networks (DFNN) Method for Damage Identification of Truss Structures Using Noisy Incomplete Modal Data," J. Build. Eng., **30**, p. 101244.
[189] Raissi, M., 2018, "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations," J. Mach. Learn. Res., **19**(1), pp. 932–955.
[190] Liu, D., Pusarla, P., and Wang, Y., 2023, "Multifidelity Physics-Constrained Neural Networks With Minimax Architecture," ASME J. Comput. Inf. Sci. Eng., **23**(3), p. 031008.
[191] Biros, G., Ghattas, O., Heinkenschloss, M., Keyes, D., Mallick, B., Tenorio, L., van Bloemen Waanders, B., Willcox, K., Marzouk, Y., and Biegler, L., 2011, *Large-Scale Inverse Problems and Quantification of Uncertainty*, G. Biros, O. Ghattas, M. Heinkenschloss, D. Keyes, B. Mallick, L. Tenorio, B. van Bloemen Waanders, K. Willcox, Y. Marzouk, and L. Biegler, eds., John Wiley & Sons, New York.
[192] Vogel, C. R., 2002, *Computational Methods for Inverse Problems*, SIAM.
[193] Franssen, H. J., Hendricks, A. A., Riva, M., Bakr, M., Van der Wiel, N., Stauffer, F., and Guadagnini, A., 2009, "A Comparison of Seven Methods for the Inverse Modelling of Groundwater Flow. Application to the Characterisation of Well Catchments," Adv. Water Resour., **32**(6), pp. 851–872.
[194] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A., 2020, "Fourier Neural Operator for Parametric Partial Differential Equations," arXiv preprint arXiv:2010.08895.
[195] Malashkhia, L., Liu, D., Lu, Y., and Wang, Y., 2023, "Physics-Constrained Bayesian Neural Network for Bias and Variance Reduction," ASME J. Comput. Inf. Sci. Eng., **23**(1), p. 011012.
[196] Randjbaran, E., Zahari, R., Vaghei, R., and Karamizadeh, F., 2015, "A Review Paper on Comparison of Numerical Techniques for Finding Approximate Solutions to Boundary Value Problems on Post-buckling in Functionally Graded Materials," Trends J. Sci. Res., **2**(1), pp. 1–6.
[197] Triebel, H., 2015, *Hybrid Function Spaces, Heat and Navier-Stokes Equations*, European Mathematical Society Press, Helsinki, Finland.
[198] Durran, D. R., 2013, *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics*, Vol. 32, Springer Science & Business Media.
[199] Prato, G. D., 2004, "The Stochastic Burgers Equation," *Kolmogorov Equations for Stochastic PDEs*, G. D. Prato, ed., Springer, Basel, pp. 131–153.
[200] Medková, D., 2018, "The Laplace Equation," *Boundary Value Problems on Bounded and Unbounded Lipschitz Domains*, Q. Du, ed., Springer, Cham, pp. 1287–1311.
[201] Genovese, L., Deutsch, T., Neelov, A., Goedecker, S., and Beylkin, G., 2006, "Efficient Solution of Poisson's Equation With Free Boundary Conditions," J. Chem. Phys., **125**(7), p. 074105.
[202] Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E., 2020, "Conservative Physics-Informed Neural Networks on Discrete Domains for Conservation Laws: Applications to Forward and Inverse Problems," Comput. Methods Appl. Mech. Eng., **365**, p. 113028.
[203] Jagtap, A. D., and Karniadakis, G. E., 2021, "Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations," AAAI Spring Symposium: MLPS, Virtual, Mar. 22–24, Vol. 10.
[204] Taghizadeh, M., Amin Nabian, M., and Alemazkoor, N., 2023, "Multi-Fidelity Physics-Informed Generative Adversarial Network for Solving Partial Differential Equations," ASME J. Comput. Inf. Sci. Eng., pp. 1–15.

[205] Mahmoudabadbozchelou, M., and Jamali, S., 2021, "Rheology-Informed Neural Networks (RhINNs) for Forward and Inverse Metamodelling of Complex Fluids," Sci. Rep., **11**(1), p. 12015.
[206] Datta, P., Pawar, N., and Faroughi, S. A., 2022, "A Physics-Informed Neural Network to Model the Flow of Dry Particles," Fall Meeting 2022, Chicago, IL, Dec. 12–16, AGU.
[207] Nguyen, L., Raissi, M., and Seshaiyer, P., 2022, "Modeling, Analysis and Physics Informed Neural Network Approaches for Studying the Dynamics of COVID-19 Involving Human-human and Human-Pathogen Interaction," Computat. Math. Biophys., **10**(1), pp. 1–17.
[208] Oommen, V., and Srinivasan, B., 2022, "Solving Inverse Heat Transfer Problems Without Surrogate Models: A Fast, Data-sparse, Physics Informed Neural Network Approach," ASME J. Comput. Inf. Sci. Eng., **22**(4), p. 041012.
[209] Dwivedi, V., and Srinivasan, B., 2022, "A Normal Equation-Based Extreme Learning Machine for Solving Linear Partial Differential Equations," ASME J. Comput. Inf. Sci. Eng., **22**(1).
[210] Dissanayake, M., and Phan-Thien, N., 1994, "Neural-Network-Based Approximations for Solving Partial Differential Equations," Commun. Numer. Methods Eng., **10**(3), pp. 195–201.
[211] Owhadi, H., 2015, "Bayesian Numerical Homogenization," Multiscale Model. Simul., **13**(3), pp. 812–828.
[212] Raissi, M., Yazdani, A., and Karniadakis, G. E., 2018, "Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data," arXiv preprint arXiv:1808.04327.
[213] Iserles, A., 2009, *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, Number 44.
[214] Nabian, M. A., Gladstone, R. J., and Meidani, H., 2021, "Efficient Training of Physics-Informed Neural Networks Via Importance Sampling," Comput. Aided Civil Infrastructure Eng., **36**(8), pp. 962–977.
[215] Yang, L., Meng, X., and Karniadakis, G. E., 2021, "B-PINNs: Bayesian Physics-Informed Neural Networks for Forward and Inverse PDE Problems With Noisy Data," J. Comput. Phys., **425**, p. 109913.
[216] Su, C. H., and Gardner, C. S., 1969, "Korteweg-de Vries Equation and Generalizations. III. Derivation of the Korteweg-de Vries Equation and Burgers Equation," J. Math. Phys., **10**(3), pp. 536–539.
[217] Constantin, P., and Foias, C., 2020, *Navier-Stokes Equations*, University of Chicago Press.
[218] Stiasny, J., Misyris, G. S., and Chatzivasileiadis, S., 2021, "Physics-Informed Neural Networks for Non-Linear System Identification for Power System Dynamics," *2021 IEEE Madrid PowerTech*, IEEE, pp. 1–6.
[219] Mao, Z., Jagtap, A. D., and Karniadakis, G. E., 2020, "Physics-Informed Neural Networks for High-Speed Flows," Comput. Methods Appl. Mech. Eng., **360**, p. 112789.
[220] Jagtap, A. D., Mao, Z., Adams, N., and Karniadakis, G. E., 2022, "Physics-Informed Neural Networks for Inverse Problems in Supersonic Flows," J. Comput. Phys., **466**, p. 111402.
[221] Zhang, T., Dey, B., Kakkar, P., Dasgupta, A, and Chakraborty, A., 2020, "Frequency-Compensated Pinns for Fluid-Dynamic Design Problems," arXiv preprint arXiv:2011.01456.
[222] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R., 2020, "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains," Adv. Neural Inf. Process Syst., **33**, pp. 7537–7547.
[223] Cheng, C., and Zhang, G.-T., 2021, "Deep Learning Method Based on Physics Informed Neural Network With Resnet Block for Solving Fluid Flow Problems," Water, **13**(4), p. 423.
[224] Lou, Q., Meng, X., and Karniadakis, G. E., 2021, "Physics-Informed Neural Networks for Solving Forward and Inverse Flow Problems Via the Boltzmann-bgk Formulation," J. Comput. Phys., **447**, p. 110676.
[225] Mahmoudabadbozchelou, M., Karniadakis, G. E., and Jamali, S., 2022, "nn-PINNs: Non-Newtonian Physics-Informed Neural Networks for Complex Fluid Modeling," Soft Matter, **18**(1), pp. 172–185.
[226] Haghighat, E., Amini, D., and Juanes, R., 2021, "Physics-Informed Neural Network Simulation of Multiphase Poroelasticity Using Stress-Split Sequential Training," Comput. Meth. Appl. Mech. Eng., **397**, p. 115141.
[227] Almajid, M. M., and Abu-Al-Saud, M. O., 2022, "Prediction of Porous Media Fluid Flow Using Physics Informed Neural Networks," J. Petroleum Sci. Eng., **208**, p. 109205.
[228] Depina, I., Jain, S., Mar, V. S., and Gotovac, H., 2022, "Application of Physics-Informed Neural Networks to Inverse Problems in Unsaturated Groundwater Flow," Georisk: Assessment Manag. Risk Engineered Syst. Geohazards, **16**(1), pp. 21–36.
[229] Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., and Barajas-S, D., 2018, "Learning Parameters and Constitutive Relationships With Physics Informed Deep Neural Networks," arXiv preprint arXiv:1808.03398.
[230] Thakur, S., Raissi, M., and Ardekani, A. M., 2022, "Viscoelasticnet: A Physics Informed Neural Network Framework for Stress Discovery and Model Selection," arXiv preprint arXiv:2209.06972.
[231] Fernandes, C., Faroughi, S. A., Ribeiro, R., Isabel, A., and McKinley, G. H., 2022, "Finite Volume Simulations of Particle-Laden Viscoelastic Fluid Flows: Application to Hydraulic Fracture Processes," Eng. Comput., pp. 1–27.
[232] Chiu, P., Wong, J. C., Ooi, C., Dao, M. H., and Ong, Y., 2022, "Can-PINN: A Fast Physics-Informed Neural Network Based on Coupled-Automatic–Numerical Differentiation Method," Comput. Methods Appl. Mech. Eng., **395**, p. 114909.
[233] Van Der Hoeven, J., 2004, "The Truncated Fourier Transform and Applications," Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, Santander, Spain, July 4–7, pp. 290–296.
[234] Raynaud, G., Houde, S., and Gosselin, F. P., 2022, "Modalpinn: An Extension of Physics-Informed Neural Networks With Enforced Truncated Fourier Decomposition for Periodic Flow Reconstruction Using a Limited Number of Imperfect Sensors," J. Comput. Phys., p. 111271.
[235] Oldenburg, J., Borowski, F., Öner, A., Schmitz, K., and Stiehm, M., 2022, "Geometry Aware Physics Informed Neural Network Surrogate for Solving Navier-Stokes Equation (GAPINN)," Adv. Model. Simul. Eng. Sci., **9**(1), p. 8.
[236] Wandel, N., Weinmann, M., Neidlin, M., and Klein, R., 2022, "Spline-PINN: Approaching PDEs Without Data Using Fast, Physics-Informed Hermite-Spline Cnns," Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, Feb. 7–14, Vol. 36, pp. 8529–8538.
[237] Jin, X., Cai, S., Li, H., and Karniadakis, G. E., 2021, "NSFnets (Navier-Stokes Flow Nets): Physics-Informed Neural Networks for the Incompressible Navier-Stokes Equations," J. Comput. Phys., **426**, p. 109951.
[238] Cheng, C., Xu, P., Li, Y., and Zhang, G., 2021, "Deep Learning Based on PINN for Solving 2 D0F Vortex Induced Vibration of Cylinder With High Reynolds Number," arXiv preprint arXiv:2106.01545.
[239] Eivazi, H., Tahani, M., Schlatter, P., and Vinuesa, R., 2022, "Physics-Informed Neural Networks for Solving Reynolds-Averaged Navier–Stokes Equations," Phys. Fluids, **34**(7), p. 075117.
[240] Wang, J., Wu, J., and Xiao, H., 2017, "Physics-Informed Machine Learning Approach for Reconstructing Reynolds Stress Modeling Discrepancies Based on Dns Data," Phys. Rev. Fluids, **2**(3), p. 034603.
[241] Zhang, X., Zhu, Y., Wang, J., Ju, L., Qian, Y., Ye, M., and Yang, J., 2022, "GW-PINN: A Deep Learning Algorithm for Solving Groundwater Flow Equations," Adv. Water Resour., p. 104243.
[242] Aliakbari, M., Mahmoudi, M., Vadasz, P., and Arzani, A., 2022, "Predicting High-Fidelity Multiphysics Data From Low-Fidelity Fluid Flow and Transport Solvers Using Physics-Informed Neural Networks," Int. J. Heat Fluid Flow, **96**, p. 109002.
[243] Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., and Barajas-Solano, D., 2020, "Physics-Informed Deep Neural Networks for Learning Parameters and Constitutive Relationships in Subsurface Flow Problems," Water Resour. Res., **56**(5), p. e2019WR026731.
[244] Kashefi, Ali, and Mukerji, Tapan, 2022, "Prediction of Fluid Flow in Porous Media by Sparse Observations and Physics-Informed Pointnet," Neural Netw., 167, pp. 80–91.
[245] Kissas, G., Yang, Y., Hwuang, E., Witschey, W. R., Detre, J. A., and Perdikaris, P., 2020, "Machine Learning in Cardiovascular Flows Modeling: Predicting Arterial Blood Pressure From Non-Invasive 4D Flow Mri Data Using Physics-Informed Neural Networks," Comput. Methods Appl. Mech. Eng., **358**, p. 112623.
[246] Arzani, A., Wang, J-X., and D'Souza, R. M., 2021, "Uncovering Near-Wall Blood Flow From Sparse Data With Physics-Informed Neural Networks," Phys. Fluids, **33**(7), p. 071905.
[247] Jagtap, A. D., Mitsotakis, D., and Karniadakis, G. E., 2022, "Deep Learning of Inverse Water Waves Problems Using Multi-fidelity Data: Application to Serre–Green–Naghdi Equations," Ocean Eng., **248**, p. 110775.
[248] Kashefi, A., and Mukerji, T., 2022, "Physics-Informed Pointnet: A Deep Learning Solver for Steady-State Incompressible Flows and Thermal Fields on Multiple Sets of Irregular Geometries," J. Comput. Phys., **468**, p. 111510.
[249] Li, M., and McComb, C., 2022, "Using Physics-Informed Generative Adversarial Networks to Perform Super-Resolution for Multiphase Fluid Simulations," ASME J. Comput. Inf. Sci. Eng., **22**(4), p. 044501.
[250] Janssen, J., Haikal, G., DeCarlo, E., Hartnett, M., and Kirby, M., 2023, "A Physics-Informed General Convolutional Network for the Computational Modeling of Materials With Damage," ASME J. Comput. Inf. Sci. Eng., pp. 1–67.
[251] Manyar, O. M., Cheng, J., Levine, R., Krishnan, V., Barbič, J., and Gupta, S. K., 2023, "Physics Informed Synthetic Image Generation for Deep Learning-Based Detection of Wrinkles and Folds," ASME J. Comput. Inf. Sci. Eng., **23**(3), p. 030903.
[252] Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R., 2020, "A Deep Learning Framework for Solution and Discovery in Solid Mechanics: Linear Elasticity," arXiv preprint arXiv:2003.02751.
[253] Shukla, K., Jagtap, A. D., Blackshire, J. L., Sparkman, D., and Karniadakis, G. E., 2021, "A Physics-Informed Neural Network for Quantifying the Microstructural Properties of Polycrystalline Nickel Using Ultrasound Data: A Promising Approach for Solving Inverse Problems," IEEE Signal Process. Mag., **39**(1), pp. 68–77.
[254] Henkes, A., Wessels, H., and Mahnken, R., 2022, "Physics Informed Neural Networks for Continuum Micromechanics," Comput. Methods Appl. Mech. Eng., **393**, p. 114790.
[255] Zhang, Z., and Gu, G. X., 2021, "Physics-Informed Deep Learning for Digital Materials," Theor. Appl. Mech. Lett., **11**(1), p. 100220.
[256] Rao, C., Sun, H., and Liu, Y., 2020, "Physics Informed Deep Learning for Computational Elastodynamics Without Labeled Data," J. Eng. Mech., **147**(8), p. 04021043.
[257] Zhou, T., Jiang, S., Han, T., Zhu, S.-P., and Cai, Y., 2023, "A Physically Consistent Framework for Fatigue Life Prediction Using Probabilistic Physics-Informed Neural Network," Int. J. Fatigue, **166**, p. 107234.
[258] Fang, Z., and Zhan, J., 2019, "Deep Physical Informed Neural Networks for Metamaterial Design," IEEE Access, **8**, pp. 24506–24513.

[259] Lax, M., and Nelson, D. F., 1976, "Maxwell Equations in Material Form," Phys. Rev. B, **13**(4), p. 1777.
[260] Zhang, E., Yin, M., and Karniadakis, G. E., 2020, "Physics-Informed Neural Networks for Nonhomogeneous Material Identification in Elasticity Imaging," arXiv preprint arXiv:2009.04525.
[261] Abueidda, D. W., Koric, S., Guleryuz, E., and Sobh, N. A., 2022, "Enhanced Physics-Informed Neural Networks for Hyperelasticity," Int. J. Numer. Meth. Eng., 124(7), pp. 1585–1601.
[262] Abueidda, D. W., Koric, S., Al-Rub, R. A., Parrott, C. M., James, K. A., and Sobh, N. A., 2022, "A Deep Learning Energy Method for Hyperelasticity and Viscoelasticity," Europ. J. Mechanics-A/Solids, **95**, p. 104639.
[263] Yuan, L., Ni, Y., Deng, X., and Hao, S., 2022, "A-PINN: Auxiliary Physics Informed Neural Networks for Forward and Inverse Problems of Nonlinear Integro-Differential Equations," J. Comput. Phys., **462**, p. 111260.
[264] Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E., 2021, "DeepXDE: A Deep Learning Library for Solving Differential Equations," SIAM Rev., **63**(1), pp. 208–228.
[265] Arora, R., 2022, "PhySRNet: Physics Informed Super-Resolution Network for Application in Computational Solid Mechanics," 2022 IEEE/ACM International Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S), IEEE, pp. 13–18.
[266] Chen, H., and Chan, W., 2020, "Higher-Order PeridynamicMaterial Correspondence Models for Elasticity," J. Elast., **142**(1), pp. 135–161.
[267] Haghighat, E., Bekar, A. C., Madenci, E., and Juanes, R., 2021, "A Nonlocal Physics-Informed Deep Learning Framework Using the Peridynamic Differential Operator," Comput. Methods Appl. Mech. Eng., **385**, p. 114012.
[268] Huang, G., Zhu, Q., and Siew, C., 2006, "Extreme Learning Machine: Theory and Applications," Neurocomputing, **70**(1–3), pp. 489–501.
[269] Yan, C. A., Vescovini, R., and Dozio, L., 2022, "A Framework Based on Physics-Informed Neural Networks and Extreme Learning for the Analysis of Composite Structures," Comput. Struct., **265**, p. 106761.
[270] Rezaei, S., Harandi, A., Moeineddin, A., Xu, B., and Reese, S., 2022, "A Mixed Formulation for Physics-Informed Neural Networks as a Potential Solver for Engineering Problems in Heterogeneous Domains: Comparison With Finite Element Method," Comput. Meth. Appl. Mech. Eng., **401**, p. 115616.
[271] Mallampati, A., and Almekkawy, M., 2021, "Measuring Tissue Elastic Properties Using Physics Based Neural Networks," 2021 IEEE UFFC Latin America Ultrasonics Symposium (LAUS), Virtual, Oct. 4–5, IEEE, pp. 1–4.
[272] Li, W., Bazant, M. Z., and Zhu, J., 2021, "A Physics-Guided Neural Network Framework for Elastic Plates: Comparison of Governing Equations-Based and Energy-Based Approaches," Comput. Methods Appl. Mech. Eng., **383**, p. 113933.
[273] Vahab, M., Haghighat, E., Khaleghi, M., and Khalili, N., 2022, "A Physics Informed Neural Network Approach to Solution and Identification of Biharmonic Equations of Elasticity," J. Eng. Mech., **148**(2), p. 04021154.
[274] Raj, M., Kumbhar, P., and Annabattula, R. K., 2021, "Physics-Informed Neural Networks for Solving Thermo-Mechanics Problems of Functionally Graded Material," arXiv preprint arXiv:2111.10751.
[275] Zhang, E., Dao, M., Karniadakis, G. E., and Suresh, S., 2022, "Analyses of Internal Structures and Defects in Materials Using Physics-Informed Neural Networks," Sci. Adv., **8**(7), p. eabk0644.
[276] Bastek, J., and Kochmann, D. M., 2022, "Physics-Informed Neural Networks for Shell Structures," Eur. J. Mech.-A/Solids, **97**, p. 104849.
[277] Katsikis, D., Muradova, A. D., and Stavroulakis, G. E., 2022, "A Gentle Introduction to Physics Informed Neural Networks, With Applications in Static Rod and Beam Problems," J. Adv. App. Comput. Math., **9**, pp. 103–128.
[278] Zhang, X., Gong, J., and Xuan, F, 2021, "A Physics-Informed Neural Network for Creep-fatigue Life Prediction of Components at Elevated Temperatures," Eng. Fract. Mech., **258**, p. 108130.
[279] Dourado, A., and Viana, F. A. C., 2020, "Physics-Informed Neural Networks for Missing Physics Estimation in Cumulative Damage Models: A Case Study in Corrosion Fatigue," ASME ASME J. Comput. Inf. Sci. Eng., **20**(6), p. 061007.
[280] Haghighat, E., Can Baker, A., Madenci, E., and Juanes, R., 2021, "Deep Learning for Solution and Inversion of Structural Mechanics and Vibrations," *Modeling and Computation in Vibration Problems, Volume 2: Soft Computing and Uncertainty*, IOP Publishing, Bristol, UK, p. 1.
[281] Zheng, B., Li, T., Qi, H., Gao, L., Liu, X., and Yuan, L., 2022, "Physics-Informed Machine Learning Model for Computational Fracture of Quasi-Brittle Materials Without Labelled Data," Int. J. Mech. Sci., **223**, p. 107282.
[282] Arora, R., Kakkar, P., Dey, B., and Chakraborty, A., 2022, "Physics-Informed Neural Networks for Modeling Rate- and Temperature-Dependent Plasticity," arXiv preprint arXiv:2201.08363.
[283] Salvati, E., Tognan, A., Laurenti, L., Pelegatti, M., and De Bona, F., 2022, "A Defect-Based Physics-Informed Machine Learning Framework for Fatigue Finite Life Prediction in Additive Manufacturing," Mater. Des., **222**, p. 111089.
[284] Bai, J., Jeong, H., Batuwatta, C. P., Xiao, S., Wang, Q., Rathnayaka, C. M., Alzubaidi, L., Liu, G., and Gu, Y., 2022, "An Introduction to Programming Physics-Informed Neural Network-Based Computational Solid Mechanics," arXiv preprint arXiv:2210.09060.
[285] Dwivedi, V., and Srinivasan, B., 2020, "Solution of Biharmonic Equation in Complicated Geometries With Physics Informed Extreme Learning Machine," ASME J. Comput. Inf. Sci. Eng., **20**(6), p. 061004.
[286] Dwivedi, V., Parashar, N., and Srinivasan, B., 2021, "Distributed Learning Machines for Solving Forward and Inverse Problems in Partial Differential Equations," Neurocomputing, **420**, pp. 299–316.
[287] Shin, Y., Darbon, J., and Karniadakis, G. E., 2020, "On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs," arXiv preprint arXiv:2004.01806.
[288] Fuks, O., and Tchelepi, H. A., 2020, "Limitations of Physics Informed Machine Learning for Nonlinear Two-Phase Transport in Porous Media," J. Mach. Learn. Model. Comput., **1**(1).
[289] Wang, S., Teng, Y., and Perdikaris, P., 2021, "Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks," SIAM J. Sci. Comput., **43**(5), pp. A3055–A3081.
[290] Wang, S., Yu, X., and Perdikaris, P., 2022, "When and Why Pinns Fail to Train: A Neural Tangent Kernel Perspective," J. Comput. Phys., **449**, p. 110768.
[291] Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W., 2021, "Characterizing Possible Failure Modes in Physics-Informed Neural Networks," Adv. Neural Inf. Process Syst., **34**, pp. 26548–26560.
[292] Weng, Y., and Zhou, D., 2022, "Multiscale Physics-Informed Neural Networks for Stiff Chemical Kinetics," J. Phys. Chem. A, **126**(45), pp. 8534–8543.
[293] Tang, K., Wan, X., and Yang, C., 2023, "Das-pinns: A Deep Adaptive Sampling Method for Solving High-Dimensional Partial Differential Equations," J. Comput. Phys., **476**, p. 111868.
[294] Sharma, P., Evans, L., Tindall, M., and Nithiarasu, P., 2023, "Stiff-PDEs and Physics-Informed Neural Networks," Arch. Comput. Methods Eng., pp. 1–30.
[295] Ji, W., Qiu, W., Shi, Z., Pan, S., and Deng, S., 2021, "Stiff-PINN: Physics-Informed Neural Network for Stiff Chemical Kinetics," J. Phys. Chem. A, **125**(36), pp. 8098–8106.
[296] Wang, Sifan, Teng, Yujun, and Perdikaris, Paris, 2020, "Understanding and Mitigating Gradient Pathologies in Physics-Informed Neural Networks," SIAM J. Sci. Comput., **43**(5), pp. A3055–A3081.
[297] Long, Y., She, X., and Mukhopadhyay, S., 2018, "Hybridnet: Integrating Model-Based and Data-Driven Learning to Predict Evolution of Dynamical Systems," Conference on Robot Learning, Zürich, Switzerland, Oct. 29–31, PMLR, pp. 551–560.
[298] Zhu, Y., Zabaras, N., Koutsourelakis, P., and Perdikaris, P., 2019, "Physics-Constrained Deep Learning for High-Dimensional Surrogate Modeling and Uncertainty Quantification Without Labeled Data," J. Comput. Phys., **394**, pp. 56–81.
[299] Geneva, N., and Zabaras, N., 2020, "Modeling the Dynamics of Pde Systems With Physics-Constrained Deep Auto-Regressive Networks," J. Comput. Phys., **403**, p. 109056.
[300] Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R., 2020, "Towards Physics-Informed Deep Learning for Turbulent Flow Prediction," Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, July 6–10, pp. 1457–1466.
[301] Ranade, R., Hill, C., and Pathak, J., 2021, "Discretizationnet: A Machine-Learning Based Solver for Navier–Stokes Equations Using Finite Volume Discretization," Comput. Methods Appl. Mech. Eng., **378**, p. 113722.
[302] Gao, H., Sun, L., and Wang, J., 2021, "Phygeonet: Physics-Informed Geometry-Adaptive Convolutional Neural Networks for Solving Parameterized Steady-State PDEs on Irregular Domain," J. Comput. Phys., **428**, p. 110079.
[303] Rao, C., Ren, P., Liu, Y., and Sun, H., "Discovering Nonlinear PDEs from Scarce Data With Physics-Encoded Learning," arXiv preprint arXiv:2201.12354.
[304] Wang, J., 1994, "A Deterministic Annealing Neural Network for Convex Programming," Neural Netw., **7**(4), pp. 629–641.
[305] Rangarajan, A., Gold, S., and Mjolsness, E., 1996, "A Novel Optimizing Network Architecture With Applications," Neural Comput., **8**(5), pp. 1041–1060.
[306] Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S., 2020, "Lagrangian Neural Networks," arXiv preprint arXiv:2003.04630.
[307] Allen-Blanchette, C., Veer, S., Majumdar, A., and Leonard, N. E., 2020, "Lagnetvip: A Lagrangian Neural Network for Video Prediction," arXiv preprint arXiv:2010.12932.
[308] Chen, Z., Zhang, J., Arjovsky, M., and Bottou, L., 2019, "Symplectic Recurrent Neural Networks," arXiv preprint arXiv:1909.13334.
[309] DiPietro, D., Xiong, S., and Zhu, B., 2020, "Sparse Symplectically Integrated Neural Networks," Adv. Neural Inf. Process Syst., **33**, pp. 6074–6085.
[310] Trask, N., Huang, A., and Hu, X., 2022, "Enforcing Exact Physics in Scientific Machine Learning: A Data-Driven Exterior Calculus on Graphs," J. Comput. Phys., **456**, p. 110969.
[311] Chen, J., and Liu, Y., 2021, "Physics-Guided Machine Learning for Multi-Factor Fatigue Analysis and Uncertainty Quantification," AIAA Scitech 2021 Forum, Virtual Event, Jan. 11–21, p. 1242.
[312] Chen, J., and Liu, Y., 2021, "Probabilistic Physics-Guided Machine Learning for Fatigue Data Analysis," Expert Syst. Appl., **168**, p. 114316.
[313] Chen, J., and Liu, Y., 2021, "Fatigue Property Prediction of Additively Manufactured Ti-6Al-4V Using Probabilistic Physics-Guided Learning," Addit. Manuf., **39**, p. 101876.
[314] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W., 2015, "Convolutional Lstm Network: A Machine Learning Approach for Precipitation Nowcasting," Adv. Neural Inf. Process., **28**.
[315] He, K., Zhang, X., Ren, S., and Sun, J., 2016, "Deep Residual Learning for Image Recognition," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, June 27–30 , pp. 770–778.
[316] Ren, P., Rao, C., Liu, Y., Wang, J., and Sun, Hao, 2022, "Phycrnet: Physics-Informed Convolutional-Recurrent Network for Solving Spatiotemporal PDEs," Comput. Methods Appl. Mech. Eng., **389**, p. 114399.

[317] Baydin, A. G., Pearlmutter, B. A, Radul, A. A., and Siskind, J. M., 2018, "Automatic Differentiation in Machine Learning: A Survey," J. Mach. Learn. Res., **18**, pp. 1–43.
[318] Rackauckas, C., Innes, M., Ma, Y., Bettencourt, J., White, L., and Dixit, V., 2019, "DiffEqFlux.jl-A Julia Library for Neural Differential Equations," arXiv preprint arXiv:1902.02376.
[319] Pontryagin, L. S., 1987, *Mathematical Theory of Optimal Processes*, CRC Press, Boca Raton, FL.
[320] Ma, Y., Dixit, V., Innes, M. J., Guo, X., and Rackauckas, C., 2021, "A Comparison of Automatic Differentiation and Continuous Sensitivity Analysis for Derivatives of Differential Equation Solutions," 2021 IEEE High Performance Extreme Computing Conference (HPEC), Virtual, Sept. 20–24, IEEE, pp. 1–9.
[321] Poli, M., Massaroli, S., Yamashita, A., Asama, H., and Park, J., 2020, "Torchdyn: A Neural Differential Equations Library," arXiv preprint arXiv:2009.09346.
[322] Lai, Z., Mylonas, C., Nagarajaiah, S., and Chatzi, E., 2021, "Structural Identification With Physics-Informed Neural Ordinary Differential Equations," J. Sound Vib., **508**, p. 116196.
[323] Roehrl, M. A., Runkler, T. A., Brandtstetter, V., Tokic, M., and Obermayer, S., 2020, "Modeling System Dynamics With Physics-Informed Neural Networks Based on Lagrangian Mechanics," IFAC-PapersOnLine, **53**(2), pp. 9195–9200.
[324] Dulny, A., Hotho, A., and Krause, A., 2021, "Neuralpde: Modelling Dynamical Systems from Data," German Conference on Artificial Intelligence, Kunstliche Intelligenz, Springer International Publishing, Cham, pp. 75–89.
[325] He, K., Zhang, X., Ren, S., and Jian, Sun, 2016, "Identity Mappings in Deep Residual Networks," European Conference on Computer Vision, Amsterdam, The Netherlands, Oct. 11–14, Springer, pp. 630–645.
[326] Goswami, S., Anitescu, C., Chakraborty, S., and Rabczuk, T., 2020, "Transfer Learning Enhanced Physics Informed Neural Network for Phase-Field Modeling of Fracture," Theor. Appl. Fract. Mec., **106**, p. 102447.
[327] Bhattacharya, K., Hosseini, B., Kovachk, N. B., and Stuart, A. M., 2020, "Model Reduction and Neural Networks for Parametric PDEs," SMAI J. Comput. Math., **7**, pp. 121–157.
[328] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, Burigede, Bhattacharya, K., Stuart, A., and Anandkumar, A., 2020, "Neural Operator: Graph Kernel Network for Partial Differential Equations," arXiv preprint arXiv:2003.03485.
[329] Migus, L., Yin, Y., Ahmed Mazari, J., and Gallinari, P., 2022, "Multi-Scale Physical Representations for Approximating PDE Solutions With Graph Neural Operators," arXiv preprint arXiv:2206.14687.
[330] Chen, T., and Chen, H., 1995, "Universal Approximation to Nonlinear Operators by Neural Networks With Arbitrary Activation Functions and Its Application to Dynamical Systems," IEEE Trans. Neural Netw., **6**(4), pp. 911–917.
[331] Lin, C., Li, Z., Lu, L., Cai, S., Maxey, M., and Karniadakis, G. E., 2021, "Operator Learning for Predicting Multiscale Bubble Growth Dynamics," J. Chem. Phys., **154**(10), p. 104118.
[332] Oommen, V., Shukla, K., Goswami, S., Dingreville, R., and Karniadakis, G. E., 2022, "Learning Two-Phase Microstructure Evolution Using Neural Operators and Autoencoder Architectures," npj Comput. Mater., **8**(1), p. 190.
[333] Goswami, S., Yin, M., Yu, Y., and Karniadakis, G. E., 2022, "A Physics-Informed Variational Deeponet for Predicting Crack Path in Quasi-Brittle Materials," Comput. Methods Appl. Mech. Eng., **391**, p. 114587.
[334] DeVore, R. A., 2017, "The Theoretical Foundation of Reduced Basis Methods," *Model Reduction and Approximation: Theory and Algorithms*, A. Cohen, K. Willcox, M. Ohlberger, and P. Benner, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, vol. 15, p. 137.
[335] Zhu, Y., and Zabaras, N., 2018, "Bayesian Deep Convolutional Encoder–Decoder Networks for Surrogate Modeling and Uncertainty Quantification," J. Comput. Phys., **366**, pp. 415–447.
[336] Grady, T. J., Khan, R., Louboutin, M., Yin, Z., Witte, P. A., Chandra, R., Hewett, R. J., and Herrmann, F. J., 2022, "Towards Large-Scale Learned Solvers for Parametric PDEs With Model-Parallel Fourier Neural Operators," arXiv preprint arXiv:2204.01205.
[337] Bui, M., Adjiman, C. S., Bardow, A., Anthony, E. J., Boston, A., Brown, S., Fennell, P. S., Fuss, S., Galindo, A., Hackett, L. A., et al. 2018, "Carbon Capture and Storage (CCS): The Way Forward," Energy Environ. Sci., **11**(5), pp. 1062–1176.
[338] Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M., 2022, "U-FNO—An Enhanced Fourier Neural Operator-Based Deep-Learning Model for Multiphase Flow," Adv. Water Resour., **163**, p. 104180.
[339] You, H., Zhang, Q., Ross, C. J., Lee, C-H., and Yu, Y., 2022, "Learning Deep Implicit Fourier Neural Operators (IFNOs) With Applications to Heterogeneous Material Modeling," Comput. Meth. Appl. Mech. Eng., **398**, p. 115296.
[340] Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A., 2021, "Neural Operator: Learning Maps Between Function Spaces," arXiv preprint arXiv:2108.08481.
[341] Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., and Karniadakis, G. E., 2022, "A Comprehensive and Fair Comparison of Two Neural Operators (With Practical Extensions) Based on Fair Data," Comput. Methods Appl. Mech. Eng., **393**, p. 114778.
[342] Ashiqur Rahman, M., Florez, M. A, Anandkumar, A., Ross, Z. E., and Azizzadenesheli, K., 2022, "Generative Adversarial Neural Operators," arXiv preprint arXiv:2205.03017.
[343] Panaretos, V. M., and Zemel, Y., 2019, "Statistical Aspects of Wasserstein Distances," Annu. Rev. Stat. Appl., **6**, pp. 405–431.
[344] Zhu, M., Zhang, H., Jiao, A., Karniadakis, G. E., and Lu, L., 2023, "Reliable Extrapolation of Deep Neural Operators Informed by Physics or Sparse Observations," Comput. Methods Appl. Mech. Eng., **412**, p. 116064.
[345] Howard, A. A, Perego, M., Karniadakis, G. E., and Stinis, P., 2023, "Multifidelity Deep Operator Networks for Data-Driven and Physics-Informed Problems," J. Comput. Phys., **493**, p. 112462.
[346] Lin, G., Moya, C., and Zhang, Z., 2023, "Learning the Dynamical Response of Nonlinear Non-autonomous Dynamical Systems With Deep Operator Neural Networks," Eng. Appl. Artif. Intell., **125**, p. 106689.
[347] You, H., Yu, Y., D'Elia, M., Gao, T., and Silling, S., 2022, "Nonlocal Kernel Network (NKN): A Stable and Resolution-Independent Deep Neural Network," J. Comput. Phys., **469**, p. 111536.
[348] Garg, S., and Chakraborty, S., 2023, "VB-DeepONet: A Bayesian Operator Learning Framework for Uncertainty Quantification," Eng. Appl. Artif. Intell., **118**, p. 105685.
[349] Yang, Y., Kissas, G., and Perdikaris, P., 2022, "Scalable Uncertainty Quantification for Deep Operator Networks Using Randomized Priors," Comput. Methods Appl. Mech. Eng., **399**, p. 115399.
[350] Lin, G., Moya, C., and Zhang, Z., 2023, "B-DeepONet: An Enhanced Bayesian DeepONet for Solving Noisy Parametric PDEs Using Accelerated Replica Exchange SGLD," J. Comput. Phys., **473**, p. 111713.
[351] Molina, A. M., Avelino, I. F., Morales, E. F., and Sucar, L. E., 2020, "Causal Based Q-Learning," Res. Comput. Sci., **149**(3), pp. 95–104.