Artificial Intelligence Review (2025) 58:323
https://doi.org/10.1007/s10462-025-11322-7

Physics-informed neural networks for PDE problems: a comprehensive review

Kuang Luo[1] · Jingshang Zhao[1] · Yingping Wang[1] · Jiayao Li[1] · Junjie Wen[1] · Jiong Liang[1] · Henry Soekmadji[4] · Shaolin Liao[1,2,3]

Accepted: 2 July 2025 / Published online: 24 July 2025

Abstract

As AI for Science continues to grow, Physics-informed neural networks (PINNs) have emerged as a transformative approach within the realm of scientific computing and deep learning, offering a robust and flexible framework for solving partial differential equations (PDEs) and other complex physical systems. By embedding physical laws directly into the architecture of neural networks, PINNs enable the integration of domain-specific knowledge, ensuring that the models adhere to known physics while fitting available data. In this paper, we provide a comprehensive overview of the state-of-the-art advancements and applications of PINNs across a broad spectrum of PDE problems. In particular, focus is given on the PINN architectures, data resampling methods for PINN, loss and activation functions, feature embedding methods and so on. What's more, the potential future directions and the anticipated evolution of PINNs are also discussed. We aim to provide valuable insights into PINNs for PDE problems, with hope to encourage further exploration and research in this promising area.

Keywords AI for science · Physics-informed neural networks · Deep learning · Partial differential equations (PDEs)

✉ Shaolin Liao
liaoshlin@mail.sysu.edu.cn; sliao5@iit.edu; saliao@purdue.edu

1 School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, Guangdong, China

2 Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA

3 Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA

4 Collins Aerospace, Rockford, IL, USA

1 Introduction

In recent decades, advancements in computer technology have profoundly transformed the landscape of scientific research. Traditionally, researchers relied on theoretical deduction and experimental verification to investigate natural phenomena. However, the emergence of computational methods has introduced various numerical simulation techniques, enabling researchers to gain deeper insights into complex real-world systems. These computer-based approaches facilitate the exploration and analysis of intricate phenomena that were previously challenging to study using conventional methods. Consequently, computational simulations have become essential tools in the scientific toolkit, enhancing our understanding across diverse disciplines.

Differential equations (DE) play a crucial role in various scientific and engineering disciplines (Boussange et al. 2023; Mallikarjunaiah 2023; Namaki et al. 2023; Soldatenko and Yusupov 2017; Rodkina and Kelly 2011) due to their ability to model complex phenomena involving functions of multiple variables. Their importance stems from the fact that many physical processes, such as heat conduction, fluid dynamics, and wave propagation, can be described by PDEs, which capture the relationships between functions and their derivatives. Depending on the number of independent variables, differential equations can be divided into the following two categories (Taylor et al. 2023), Ordinary differential equation (ODE) involves only one independent variable and its derivative,

$$\frac{du}{dx} + u = 0. \tag{1}$$

Partial differential equation (PDE) involves multiple independent variables and their partial derivatives,

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2}. \tag{2}$$

Different from algebraic equations, differential equations express the equational relationship between unknown functions and their derivatives.

In practical applications, the solving of PDEs has gained immense importance in fields such as aerospace engineering, where they are utilized to optimize aircraft designs and analyze airflow around structures. Additionally, in meteorology, PDEs play a crucial role in weather prediction models, enabling accurate forecasts by simulating atmospheric dynamics. The ability to effectively solve PDEs is fundamental to advancing technology and improving predictive capabilities across various disciplines. The mathematical theory of PDEs has developed significantly, leading to various methods for their analysis and solution. To solve some simplified partial differential equations, common operators can be used (Melchers et al. 2023). At present, the more common methods for solving partial differential equations are the Finite Element Method (FEM) (Liu et al. 2022; Chernyshenko and Olshanskii 2015), the Finite Volume Method (FVM) (Dick 2009), Particle-based methods (Oñate and Owen 2011), and the Finite Cell Method (FCM) (Kollmannsberger et al. 2019). FEM has proven to be a convenient and accurate approach, utilizing extensive computational resources (Innerberger and Praetorius 2023). However, the practicality of the multi-iteration solution is limited. To enhance the capability of PDE solving, researchers have explored the use of spline functions (Kolman et al. 2017; Qin et al. 2019). Similarly, FEM discretizes and numerically approximates the PDE solution, making the introduction of Fourier transforms or Laplace transforms with higher efficiency in expressing PDEs more feasible (Mugler and Scott 1988; El-Ajou 2021) (Fig. 1).

Recently, remarkable progress in deep learning has led to groundbreaking advancements in various fields, such as computer vision and natural language processing. Additionally, deep learning has had a profound impact on scientific computing. Physics-informed machine learning (PIML) has emerged as an approach that combines traditional machine learning techniques with domain-specific physical knowledge to solve complex scientific and engineering problems. Unlike conventional machine learning methods that rely solely on data, PIML integrates physical laws and constraints into the learning process, allowing models to learn not only from data but also from known governing equations, boundary conditions, and other physical principles. This hybrid approach enhances the model's ability to generalize, especially in scenarios where data is scarce, noisy, or incomplete. PIML has played an increasingly crucial role across various fields, including optimization (Mowlavi and Nabi 2023; Hwang et al. 2022; Zhang et al. 2022b), fluid (Cai et al. 2021; Di Leoni et al. 2023; Zhang et al. 2023; Ranade et al. 2021; Sun et al. 2020; Hanna et al. 2022), cybernetics (Schiassi et al. 2022; Patel et al. 2023; Zheng et al. 2023; Faria et al. 2024), climate (Li et al. 2024; Bonev et al. 2023), mechanics (Tripura and Chakraborty 2023; Wang et al. 2023; Diao et al. 2023; Le-Duc et al. 2024), physics (Pan et al. 2024; Liu et al. 2024a; Jeong et al. 2024; Jo et al. 2024), quantum computing (Norambuena et al. 2024; Xiao et al. 2024; Sedykh et al. 2024), etc.

Methods based on deep learning for solving PDE problems have been continuously studied since 1990 s (Dissanayake and Phan-Thien 1994; Lagaris et al. 1998; Kumar and Yadav 2011; Blechschmidt and Ernst 2021; Hafiz et al. 2024). Particularly, physics-informed neural networks (PINNs) have gained prominence as an efficient method for tackling both forward and inverse PDE problems (Raissi et al. 2019). By integrating the governing PDEs into

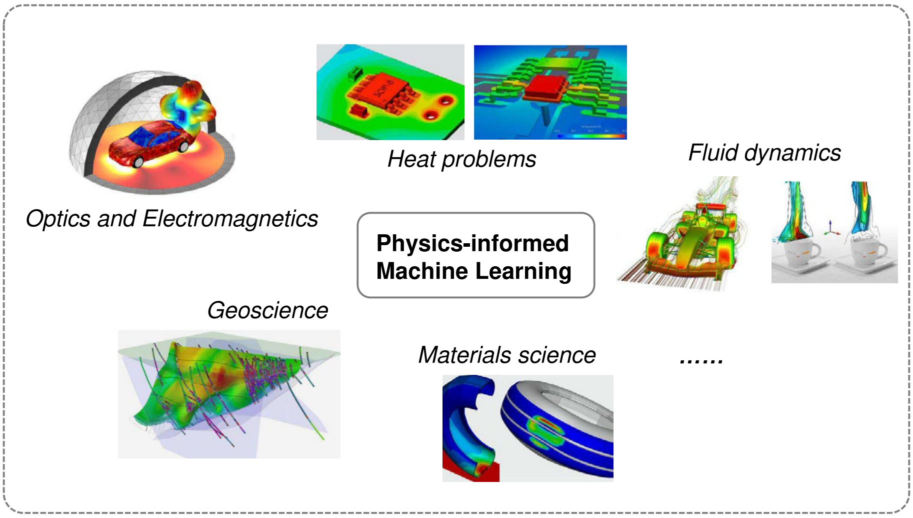
Heat problems
Fluid dynamics
Optics and Electromagnetics
Physics-informed Machine Learning
Geoscience
Materials science
......

Fig. 1 Various applications of physics-informed machine learning

the loss function, PINNs leverage automatic differentiation to guide the training process, ensuring that the learned model adheres to physical laws. This mesh-free approach provides significant flexibility, enabling PINNs to seamlessly incorporate physics-based constraints alongside observational data within a single framework. This capability makes PINNs particularly advantageous in scenarios where traditional PDE solvers may struggle, such as high-dimensional problems or those with complex geometries. Additionally, the ability to fuse data with physical models enhances predictive accuracy and robustness, allowing for more reliable simulations and solutions. As research continues, the versatility of PINNs is likely to expand, further solidifying their role in solving challenging PDE problems across various domains (Table 1).

1.1 Basics of physics-informed neural network (PINN)

1.1.1 Problem formulation

PINNs are a class of machine learning methods that integrate domain-specific knowledge, typically described by PDEs, into the neural network training process. The key idea behind PINNs is to enforce the governing equations of a physical system as constraints during the training of a neural network (Fig. 2).

In the context of solving a PDE-constrained optimization problem, the solution $u(\boldsymbol{x}, t)$ represents the the state variable, where $\boldsymbol{x}$ denotes the spatial coordinates and *t* represents time. The associated PDE can typically be written as,

$$\begin{aligned} f\left(\boldsymbol{x}, t, \frac{\partial}{\partial \boldsymbol{x}}u, \frac{\partial}{\partial t}u, \boldsymbol{\lambda}\right) &= 0,\ \boldsymbol{x} \in \Omega, t \in [0, T], \\ u(\boldsymbol{x}, 0) &= h(x),\ \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}, t) &= g(\boldsymbol{x}, t),\ \boldsymbol{x} \in \partial\Omega, t \in [0, T], \end{aligned} \tag{3}$$

where ∂ represents the differential operator, $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_n]$ represents the parameters of the PDE, and *f* denotes the residual of the PDE. Ω and $\partial\Omega$ represent the solution domain and its boundary, respectively. The initial condition is given by *h*(*x*), and the boundary condition is $g(\boldsymbol{x}, t)$.

The neural network is utilized to approximate the solution $u(\boldsymbol{x}, t)$ by $\tilde{u}(\boldsymbol{x}, t; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the neural network parameters. To incorporate this PDE into the neural network

Table 1 The notations used in the paper

Notations	Description
u	The unknown solution to the PDE problem
x	The spatial coordinate in the domain of the system
t	The time variable, typically used in time-dependent problems
$\boldsymbol{\lambda}$	The parameters of the physical system governed by the PDE
$\boldsymbol{\theta}$	The neural network parameters
Ω	The domain of the physical system
$\mathcal{L}_f$	The residual loss
$\mathcal{L}_b$	The boundary condition loss
$\mathcal{L}_i$	The initial condition loss

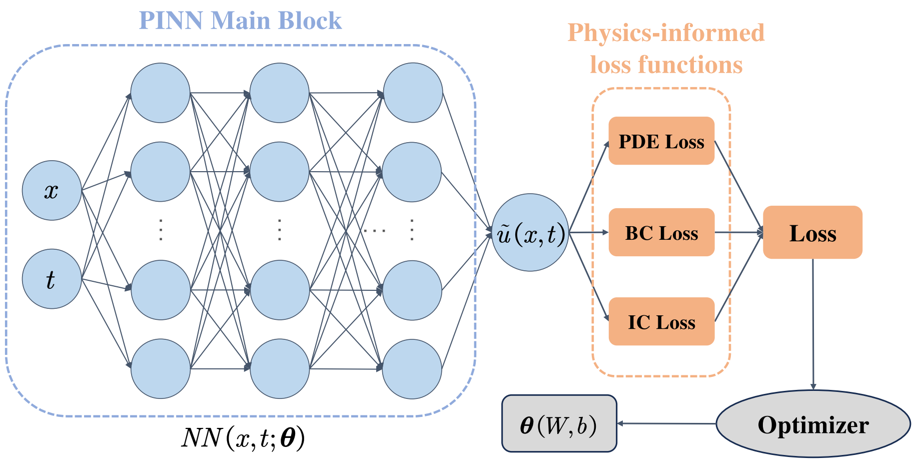
PINN Main Block
x
t
$NN(x,t;\boldsymbol{\theta})$
$\tilde{u}(x,t)$
Physics-informed loss functions
PDE Loss
BC Loss
IC Loss
Loss
Optimizer
$\boldsymbol{\theta}(W,b)$

Fig. 2 The framework of PINNs

training, we can define a loss function that consists of two terms, a data-driven term that enforces the neural network to fit the available data, and a physics-informed term that enforces the PDE constraints. The loss function can be formulated as,

$$\mathcal{L} = \mathcal{L}_{\text{physics}} + \mathcal{L}_{\text{data}}, \tag{4}$$

Here, $\mathcal{L}_{\text{physics}}$ enforces the PDE constraints as follows,

$$\mathcal{L}_{\text{physics}} = w_f \mathcal{L}_f\left(\boldsymbol{\theta}; \mathcal{T}_f\right) + w_b \mathcal{L}_b\left(\boldsymbol{\theta}; \mathcal{T}_b\right) + w_i \mathcal{L}_i\left(\boldsymbol{\theta}; \mathcal{T}_i\right), \tag{5}$$

with

$$\begin{aligned} \mathcal{L}_f\left(\boldsymbol{\theta}; \mathcal{T}_f\right) &= \frac{1}{|\mathcal{T}_f|} \sum_{\boldsymbol{x}_f \in \mathcal{T}_f} \left| f\left(\boldsymbol{x}_f, t, \boldsymbol{\lambda}\right)\right|^2, \\ \mathcal{L}_b\left(\boldsymbol{\theta}; \mathcal{T}_b\right) &= \frac{1}{|\mathcal{T}_b|} \sum_{\boldsymbol{x}_b \in \mathcal{T}_b} \left| \tilde{u}\left(\boldsymbol{x}_b, t; \theta\right) - g\left(\boldsymbol{x}_b, t\right)\right|^2, \\ \mathcal{L}_i\left(\boldsymbol{\theta}; \mathcal{T}_i\right) &= \frac{1}{|\mathcal{T}_i|} \sum_{\boldsymbol{x}_i \in \mathcal{T}_i} \left| \tilde{u}\left(\boldsymbol{x}_i, 0; \theta\right) - h\left(\boldsymbol{x}_i\right)\right|^2, \end{aligned} \tag{6}$$

where w_f, w_b and w_i are the weights; $\mathcal{T}_f$, $\mathcal{T}_b$, $\mathcal{T}_i$ are the sets of samples inside the domain, on the boundary conditions, and on the initial conditions, respectively. $\mathcal{L}_{\text{data}}$ measures the error between the predicted and observed data,

$$\mathcal{L}_{\text{data}} = \frac{1}{|\mathcal{T}_d|} \sum_{x_d \in \mathcal{T}_d} \left| \tilde{u}\left(x_d, t; \theta\right) - u\left(x_d, t\right)\right|^2, \tag{7}$$

where $\{u\left(\boldsymbol{x}_d, t\right)\}_{\boldsymbol{x}_d \in \mathcal{T}_d}$ represents the known observational data. By minimizing this combined loss function using techniques like gradient descent, PINNs can learn the underlying physics of a system directly from data, making them powerful tools for solving inverse problems and discovering hidden patterns in complex physical systems.

1.1.2 Performance evaluation for PINN

The evaluation of PINN involves several critical metrics that assess both the accuracy of the solution and the degree to which the network adheres to the underlying physical principles embedded in the governing equations. These metrics are essential for determining the effectiveness of PINN in solving complex PDEs while ensuring that the solutions maintain physical consistency and adhere to boundary and initial conditions. Below, we elaborate on the key evaluation metrics commonly used in assessing the performance of PINN.

- **L^2 relative error**
  The L^2 relative error measures the relative error in a vector space. Mathematically, the L^2 relative error between the PINN prediction $\tilde{u}\left(\boldsymbol{x},t;\boldsymbol{\theta}\right)$ and the true solution $u\left(\boldsymbol{x},t\right)$ can be expressed as follows,

$$\begin{aligned}\text{L2 error} &= \frac{\left\|\tilde{u}\left(\boldsymbol{x},t;\boldsymbol{\theta}\right)-u\left(\boldsymbol{x},t\right)\right\|_2}{\left\|u\left(\boldsymbol{x},t\right)\right\|_2}\\ &= \frac{\sqrt{\sum_{i=1}^{N}\left|\tilde{u}\left(\boldsymbol{x}_i,t_i;\boldsymbol{\theta}\right)-u\left(\boldsymbol{x}_i,t_i\right)\right|^2}}{\sqrt{\sum_{i=1}^{N}\left|u\left(\boldsymbol{x}_i,t_i\right)\right|^2}},\end{aligned} \tag{8}$$

where $\|\cdot\|_2$ denotes the L^2 norm.

- **Root mean square error (RMSE)** RMSE serves as a critical metric for quantifying the dispersion of prediction errors. It is a commonly used metric in regression and machine learning models to evaluate the accuracy of predictions. Mathematically, RMSE is defined as,

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|\tilde{u}(\boldsymbol{x}_i,t_i;\boldsymbol{\theta})-u(\boldsymbol{x}_i,t_i)\right|^2} \tag{9}$$

- **PDE residual** The PDE residual measures the extent to which the predicted solution satisfies the PDE. It represents how much the model's solution deviates from satisfying the governing equation at a given point in space and time. Mathematically, for a general PDE of the form $\mathcal{L}\left(u(\boldsymbol{x},t)\right)=f(\boldsymbol{x},t)$, where $\mathcal{L}$ is the differential operator and $f(\boldsymbol{x},t)$ is the source term of PDE. The PDE residual is represent as follows,

$$\text{PDE residual} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{L}\left(\tilde{u}(\boldsymbol{x}_i,t_i;\boldsymbol{\theta})\right)-f(\boldsymbol{x}_i,t_i)\right)^2. \tag{10}$$

1.2 Paper organization and contributions

The remainder of this paper is structured as follows. Section 2 provides a comprehensive analysis of the existing literature on PINN-based PDE solvers. Sections 3 shows the applications of PINNs and Sect. 4 introduces the current tools and codes openly for carrying out PINNs. Section 5 discusses the challenges and some possible future directions in the field. Finally, Sect. 6 concludes the paper.

This review aims to provide a comprehensive overview of PINNs, highlighting their theoretical foundations, applications, and advancements. The primary contributions of this work are as follows,

- **Introduction to PINN** The paper offers a clear introduction to the fundamental principles of PINNs, explaining how they integrate deep learning techniques with physical laws to solve PDE problems.
- **Discussion of key methodologies** The review delves into the various methodologies in PINNs, including the architectures, adaptive data resampling methods, loss function designs, feature embedding, etc.
- **Review of applications** The paper surveys the diverse applications of PINNs across multiple domains, such as fluid dynamics, material science, and biology, demonstrating their versatility in addressing real-world challenges.
- **Recent advancements** The paper summarizes recent advancements in PINN research, including improvements in computational accuracy and efficiency, generalization capabilities.
- **Emerging paradigms and future directions** The integration of PINNs with emerging paradigms like operator learning is highlighted. Additionally, the paper discusses the current challenges faced by PINNs and propose potential directions for future research to enhance their applicability and performance.

2 PINN solver for PDE problems

As early as the twentieth century, some researchers began exploring the application of machine learning methods to solve PDEs. The concept was to exploit the strong approximation capabilities of neural networks, using time and spatial locations as inputs. Through the network's mapping, the solution to the PDE could ultimately be obtained. In 1994, Dissanayake and Phan-Thien (1994) were the pioneers in employing a neural network-based approach for solving PDEs. They utilized the "universal approximator" property of neural networks and, through the collocation method, converted the numerical problem of solving PDEs into an unconstrained minimization problem.

Subsequently, Lagaris et al. (1998) introduced an artificial neural network model for solving PDEs with various boundary conditions. This approach showed significant potential in terms of solution accuracy, generalization ability, and the number of parameters needed. Monterola and Saloma (2003) introduced an unsupervised neural network-based method for solving the nonlinear Schrödinger equation and provided an upper bound estimate for the error. Aarts and van der Veer (2001) developed neural networks with a specific structure that integrated the knowledge of PDEs along with their boundary and initial conditions, while simultaneously training multiple neural networks.

However, these early methods were limited by the constraints of neural network structures and algorithms, which significantly hindered their ability to handle more complex problems.

In recent years, with the rapid development of artificial intelligence technology, high-performance computing hardware such as GPUs and TPUs have emerged, and deep learning frameworks like TensorFlow (Abadi et al. 2016) and PyTorch (Paszke et al. 2019) have been launched and continuously improved. The application of deep learning in solving PDEs has made further progress. In 2019, Raissi et al. (2019) proposed the PINN model, which pioneered a new paradigm for solving PDEs using neural networks. By integrating the PDE and its initial and boundary conditions into the loss function of the neural network, and using automatic differentiation and backpropagation to guide the training process, the model can seamlessly combine physics-based constraints with observational data within a unified framework.

Figure 3 illustrates the timeline of neural network-based methods for solving PDEs. It provides a clear and concise visual representation of the evolution of these methods, emphasizing the central and transformative role that neural networks now play in modern PDE solving approaches. The timeline highlights the progressive integration of neural networks, demonstrating their growing influence and effectiveness in addressing the complexities of PDEs (Table 2).

2.1 Architectures for PINN

Various architectures tailored to address specific challenges have emerged in PINNs. These different PINN architectures represent innovative adaptations of traditional neural network structures, specifically designed to enhance performance in solving PDE problems. This section explores and compares various PINN architectures, highlighting their unique design principles, computational efficiencies, and applicability across diverse domains. The main architectures covered in PINNs mainly include Multi-layer perceptron (MLP) (Hornik et al. 1989), Convolutional neural networks (CNN) (Krizhevsky et al. 2012; Gu et al. 2018), Recurrent neural networks (RNN) (Giles et al. 1994; Schuster and Paliwal 1997), and others such as Generative adversarial networks (GAN) (Goodfellow et al. 2014), Transformer (Vaswani et al. 2017).

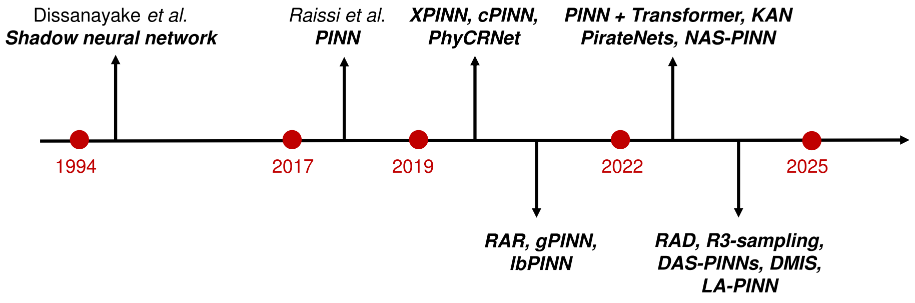
Dissanayake *et al.*
Shadow neural network
Raissi et al.
PINN
XPINN, cPINN, PhyCRNet
PINN + Transformer, KAN PirateNets, NAS-PINN
1994
2017
2019
2022
2025
RAR, gPINN, lbPINN
RAD, R3-sampling, DAS-PINNs, DMIS, LA-PINN

Fig. 3 The timeline of neural network based methods for solving PDEs

Table 2 An overview of PINN solver for PDE problems

Method	Representatives	
Architectures for PINN	MLP	vanilla PINN (Raissi et al. 2019; He et al. 2020)
	CNN	PhyGeoNet (Gao et al. 2021; Fang 2022), Spline-PINN (Wandel et al. 2022), PICNN (Lei et al. 2023), f-PICNN (Yuan et al. 2024)
	RNN	PhyCRNet (Ren et al. 2022; Mavi et al. 2023)
	GAN	PI-GANs (Yang et al. 2020; Gao and Ng 2022)
	KAN	Wang et al. (2024c), Shukla et al. (2024), Rigas et al. (2024)
	Transformer	PIT (Santos et al. 2023), PINNsFormer (Zhao et al. 2024), Transolver (Wu et al. 2024)
	Other architectures for PINN	NAS-PINN (Wang and Zhong 2024), SPINN (Cho et al. 2022), sparabreak PirateNets (Wang et al. 2024b)
	Domain decomposition	XPINNs (Karniadakis and Em 2020), cPINNs (Jagtap et al. 2020a), (Wu et al. 2022)
	Activation functions for PINN	Jagtap et al. (2020b), PIKF-NNs (Fu et al. 2024), PAFs (Abbasi and Andersen 2024; Zeng et al. 2022)
Adaptive data resampling	RAR (Lu et al. 2021b; Nabian et al. 2021), RAR-D (Wu et al. 2023), R3 sampling (Daw et al. 2023) DAS-PINNs (Tang et al. 2023), DMIS (Yang et al. 2023), GAS (Jiao et al. 2024) PINNACLE (Lau et al. 2024), AAS (Tang et al. 2024)	
Loss functions for PINN	Loss reweighting	Wang et al. (2021c, 2022a), lbPINNs (Xiang et al. 2022) SelectNet (Gu et al. 2021), SA-PINNs (McClenny and Braga-Neto 2023), LA-PINN (Song et al. 2024)
	Novel loss functions	gPINNs (Yu et al. 2022; Wang et al. 2022b, 2024a,d)
Feature embedding and augmentation	Wang et al. (2021a), Li et al. (2024a, 2024c) PD-PINN (Peng et al. 2020), sf-PINN (Wong et al. 2024), DaPINN (Guan et al. 2023)	

2.1.1 Multilayer perceptron (MLP)

The multilayer perceptron (MLP) (Hornik et al. 1989) is a fundamental architecture within artificial neural networks, consisting of layers of interconnected neurons that can approximate complex non-linear functions by adjusting weights iteratively and using activation functions. At the core of its theoretical foundation lies the Universal Approximation Theorem (Cybenko 1989; Chen and Chen 1995), which states that a single hidden layer MLP can approximate any continuous function with suitable activation functions and a finite number of neurons. The input to the MLP consists of a vector $\boldsymbol{x}$ representing the input features (Fig. 4). Each element of $\boldsymbol{x}$ corresponds to a feature of the input data. For each hidden layer l, where $l = 1, 2, \ldots, L$, compute:

$$\begin{aligned} \boldsymbol{z}^{(l)} &= \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}, \\ \boldsymbol{a}^{(l)} &= \sigma^{(l)}(\boldsymbol{z}^{(l)}), \end{aligned} \tag{11}$$

where the weight matrix $\boldsymbol{W}$ and the bias vector $\boldsymbol{b}$ are parameters and $\sigma()$ is the non-linear activation function. The output layer computes the final prediction vector,

$$\boldsymbol{y} = \boldsymbol{W}^{(L+1)}\boldsymbol{a}^{(L)} + \boldsymbol{b}^{(L+1)}. \tag{12}$$

The characteristics of the fully connected feed-forward networks (FNN) architecture, also known as MLP, such as the number of layers and neurons per layer, are typically determined empirically for different PDE problems. In the vanilla PINN (Raissi et al. 2019), different typologies of FNN are ultilized, for example, a 5 layer FNN with 100 neurons per layer for the Schrodinger equation, a 4 layer FNN and 200 neurons per layer for the Allen–Cahn equation. In He et al. (2020), the effect of neural network size on prediction accuracy is investigated. It indicates that employing excessively small or large sizes for PINNs can lead

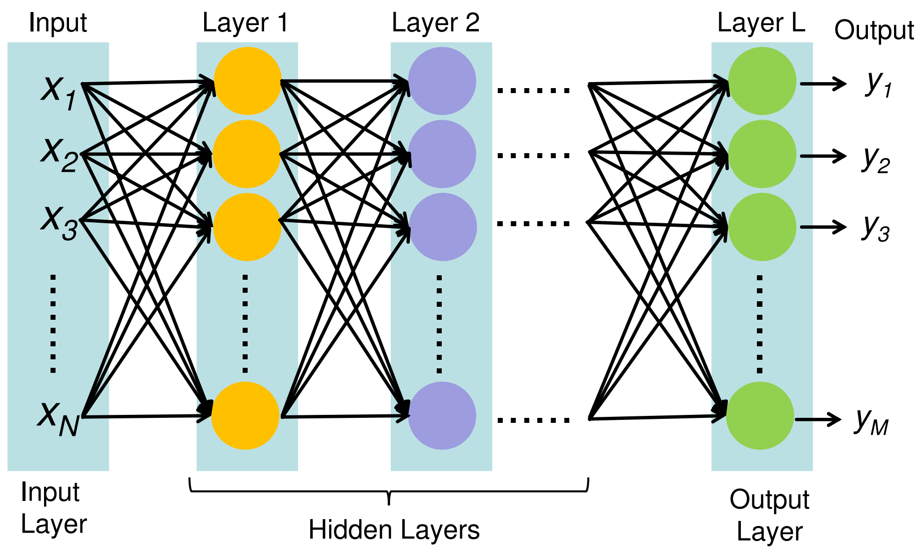

Fig. 4 The framework of MLP

to diminished performance. The theoretical analysis of how the size of neural networks in PINN affects performance across various problems remains unexplored due to their black-box nature. Addressing this question is crucial yet challenging.

2.1.2 Convolutional neural networks (CNN)

The convolutional neural network (CNN) (Li et al. 2022) is a deep learning model specifically designed to handle data with a grid-like topology, particularly image data. The core of CNNs lies in the use of convolutional layers and pooling layers to extract and compress features, making them highly effective in tasks such as image classification, object detection, and image segmentation. Convolutional layers extract local features through the convolution operation (i.e., applying a set of learnable filters), while pooling layers reduce the spatial dimensions of the feature maps via down-sampling operations, thereby reducing computational complexity and preventing overfitting. The success of CNNs is largely attributed to their ability to automatically learn hierarchical features in images, ranging from low-level features (such as edges) to high-level features (such as parts of objects).

The convolution operation involves sliding a learnable filter (or kernel) over the input data and computing the dot product between the filter and the input data, described as follows,

$$M = (I * K)(i, j) = \sum_{m} \sum_{n} I(i + m, j + n) \cdot K(m, n) \tag{13}$$

where I is the input image, K is the convolution kernel, M is the output feature map, and $*$ denotes the convolution operation. Then the pooling layer reduces the spatial dimensions of the feature maps, lowering computational complexity and preventing overfitting,

$$P(i, j) = \max_{0 \leq m < p} \max_{0 \leq n < p} M(s \cdot i + m, s \cdot j + n) \tag{14}$$

where P is the pooled output, M is the feature map, p is the pooling window size, and s is the stride.

Many researchers have explored the integration of CNN architecture into PINNs. Gao et al. (2021) proposed PhyGeoNet, a physics-informed, geometry-adaptive CNN for solving parameterized steady-state PDEs on irregular domains. It uses elliptic coordinate mapping to transform irregular geometries into regular ones, enabling the application of standard CNN architectures. PhyGeoNet can learn solutions without labeled data on Heat, Navier–Stokes, and Poisson equations. In Fang (2022), a high-efficient hybrid PINN for solving PDEs is introduced. It combines ideas from CNNs and finite volume methods, using a local fitting method to approximate the differential operator instead of automatic differentiation. Numerical experiments validate its correctness and efficiency, showing significant advantages over conventional PINNs in both computational time and accuracy. Wandel et al. (2022) introduced Spline-PINN, which combines the benefits of PINNs and CNNs using Hermite spline kernels. It enables continuous interpolation of grid-based representations suitable for CNNs, allowing training without precomputed data using a physics-informed loss. Spline-PINN offers fast, accurate solutions that generalize to unseen domains, as demonstrated with the Navier-Stokes and damped wave equations. Lei et al. (2023) proposed a physics-informed

CNN (PICNN) specifically designed for solving PDEs on the sphere. Building on recent advancements in deep learning and spherical harmonic analysis, the authors establish rigorous theoretical bounds for the approximation error of PICNNs in Sobolev norms. They develop a novel localization complexity analysis to demonstrate fast convergence rates for PICNNs. The numerical experiments highlights the PICNN's effectiveness in solving PDEs on spherical domains. In Yuan et al. (2024), a CNN based f-PICNN is designed for solving PDEs in space-time domains. Leveraging a stack of nonlinear convolutional units and a memory mechanism, f-PICNN can handle PDEs with nonlinearity, drastic gradients, and multiscale characteristics. The network is unsupervised and does not require labeled data. The efficacy of f-PICNN is demonstrated in various benchmark PDEs, showcasing its capability to accurately approximate solutions (Fig. 5).

2.1.3 Recurrent neural networks (RNN)

Recurrent Neural Networks (RNNs) (Lipton et al. 2015) are designed for processing sequential data and are characterized by their recurrent connections, which capture temporal dependencies in the data. RNNs share weights across time steps and introduce hidden states, enabling the network to remember and utilize past information. This architecture excels in tasks such as time series prediction, language modeling, and sequence labeling. The theoretical foundation of RNNs includes their recursive nature, allowing information to propagate through the sequence. Despite facing challenges like vanishing and exploding gradients during training on long sequences, RNNs and their variants [such as LSTM (Hochreiter and Schmidhuber 1997; Yu et al. 2019) and GRU (Chung et al. 2014)] have made significant progress in addressing these issues.

The basic structure of an RNN consists of an input layer, a hidden layer, and an output layer, with recurrent connections in the hidden layer. These recurrent connections allow the

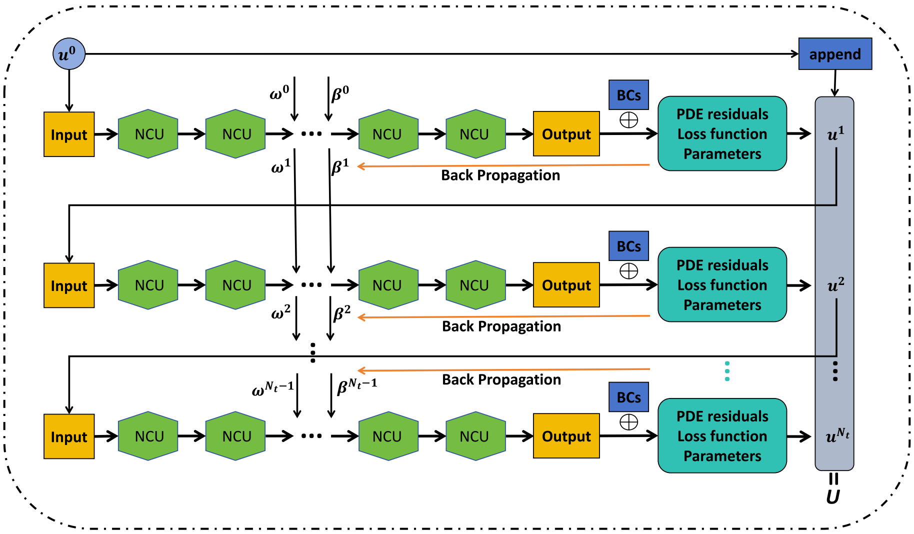

Fig. 5 The neural network architecture of the f-PICNN (Yuan et al. 2024) for time-dependent PDEs, with Nonlinear Convolutional Units (NCUs)

hidden state to be updated based on the current input and the hidden state from the previous time step. The hidden state update formula is as follows,

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \tag{15}$$

where h_t is the hidden state at time step t, x_t is the input at time step t, W_{xh} is the weight matrix from input to hidden layer, W_{hh} is the weight matrix from hidden layer to hidden layer, b_h is the bias of the hidden layer, and f is the activation function, typically tanh or ReLU. The output calculation can be formulated as

$$y_t = g(W_{hy}h_t + b_y), \tag{16}$$

where y_t is the output at time step t W_{hy} is the weight matrix from hidden layer to output layer b_y is the bias of the output layer g is the activation function (Fig. 6).

Ren et al. (2022) proposed PhyCRNet, a novel physics-informed convolutional-recurrent learning architecture designed to solve spatiotemporal PDEs without requiring labeled data. PhyCRNet utilizes an encoder-decoder convolutional LSTM network for efficient spatial feature extraction and temporal evolution. The proposed PhyCRNet is evaluated on various benchmark problems, demonstrating its ability to accurately solve PDEs and outperforms existing PINN approaches. Mavi et al. (2023) proposed an unsupervised CNN-LSTM architecture with nonlocal interactions for solving PDEs. The network evaluates derivatives of the field variable by utilizing the nonlocal peridynamic differential operator (PDDO) as a convolutional filter. The architecture captures time dynamics in a reduced latent space through encoder-decoder layers with a Convolutional LSTM layer in between. The physics is enforced at the network's output and in the latent space. Benchmark PDEs demonstrate the training performance and extrapolation capability of this novel architecture.

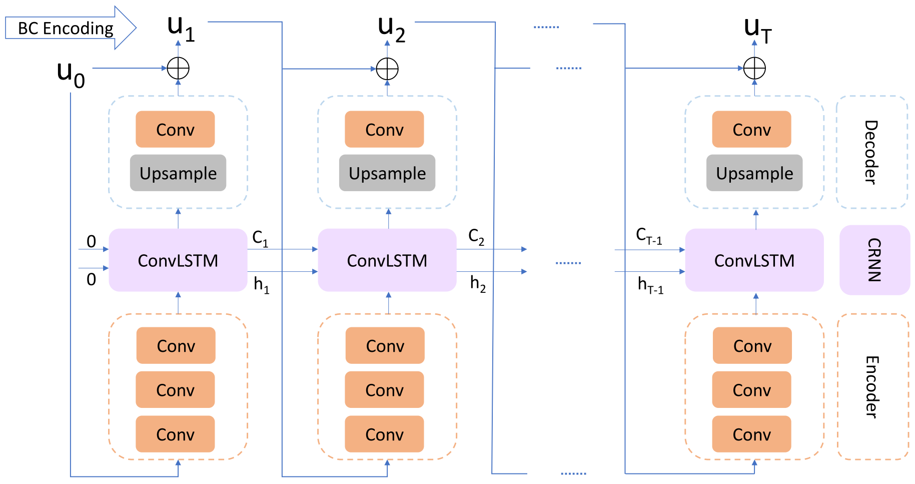

Fig. 6 The framework of PhyCRNet (Ren et al. 2022)

2.1.4 Generative adversarial networks (GAN)

GANs are a class of deep generative models designed to generate new, synthetic data samples that resemble a given set of real data. The architecture consists of two neural networks: a generator and a discriminator, which are trained simultaneously through adversarial processes. The generator creates data samples with the aim of deceiving the discriminator into believing that these samples are real, while the discriminator's goal is to distinguish between real and synthetic samples. This competitive framework fosters the generation of increasingly realistic data (Creswell et al. 2018).

The GAN framework has been integrated into PINNs in some works. Yang et al. (2020) proposed physics-informed GANs (PI-GANs) for solving forward, inverse, and mixed stochastic problems based on limited scattered measurements. PI-GANs encode physical laws in the form of stochastic differential equations (SDEs). Three generators are used for the PI-GANs and demonstrates good approximation of stochastic processes even with mismatches between the input noise and the effective dimensionality of the target processes. Gao and Ng (2022) examines a physics-informed approach for applying Wasserstein Generative Adversarial Networks (WGANs) to quantify uncertainties in the solutions of PDEs. Utilizing groupsort activation functions within the adversarial network's discriminators, the method employs network generators to learn the uncertainties present in the solutions of PDEs based on initial/boundary data. Theoretical findings indicate that, under mild conditions, the generalization error of the computed generator tends towards the network's approximation error with high probability given a sufficient number of samples. Numerical experiments have validate these theoretical insights, demonstrating the capability of the proposed model to detect uncertainties and approximate exact solutions even when noise is predominant.

2.1.5 Kolmogorov–Arnold networks (KAN)

Kolmogorov–Arnold Networks (KAN) (Liu et al. 2024b) build upon the Kolmogorov–Arnold Representation Theorem (KART). This theorem stipulates that any continuous function of multiple variables can be decomposed into a finite composition of univariate continuous functions plus addition operations. Accordingly, KAN models high-dimensional functions as a sum of univariate functions. By including learnable edge activation functions (i.e., weights), KAN boosts interpretability and accuracy. In mathematical terms, for a smooth function $f : [0, 1]^n \to \mathbb{R}$, one can write:

$$f(\mathrm{x}) = \sum_{q=1}^{2^n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right),$$

where $\mathrm{x} = (x_1, \ldots, x_n)$ denotes the input vector, and Φ_q and $\phi_{q,p}$ represent univariate functions.

In contrast, the widely adopted MLP employs the Universal Approximation Theorem (UAT) as its theoretical basis, typically placing activation functions at the neurons (i.e., nodes). Each neuron then receives inputs and produces outputs via an activation function such as ReLU, Sigmoid, or Tanh. Symbolically, an MLP might be expressed as:

$$\mathrm{MLP}(\mathrm{x}) = \big(\mathrm{W}_L\,\sigma\big(\mathrm{W}_{L-1}\ldots\sigma(\mathrm{W}_1\mathrm{x}+\mathrm{b}_1)\ldots\big)+\mathrm{b}_L\big),$$

where W_l and b_l are the weight matrix and bias of layer l, respectively, and σ is the activation function. This node-centric scheme is often considered opaque, since the internal workings are not straightforward to interpret.

KAN, on the other hand, locates its activation functions on network edges rather than nodes. Consequently, each individual input passes through a learnable univariate mapping (e.g., a B-spline function) and these outputs are subsequently combined by addition nodes. A general KAN can be written as:

$$\mathrm{KAN}(\mathrm{x}) = (\Phi_{L-1}\circ\ldots\circ\Phi_1\circ\Phi_0)(\mathrm{x}),$$

where Φ_l represents the transformation in layer l. Specifically,

$$\mathrm{x}_{l+1} = \big[\phi_{l,1,1}(\mathrm{x}_l),\ldots,\phi_{l,n_{l+1},n_l}(\mathrm{x}_l)\big],$$

with each $\phi_{l,j,i}$ denoting the univariate activation from node i to node j at layer l. This structure promotes transparency, enabling researchers to observe how univariate functions are configured and how data flows through the network.

Since their decomposition of high-dimensional tasks into smaller univariate pieces, KANs are well-suited for complex, nonlinear PDE problems. Compared to traditional MLPs, KANs permit clearer visualization of their internal function compositions, making it easier to refine specific parts of the model for improved PDE solutions. Additionally, KAN can derive symbolic formulas from its learned representations, aiding the validation of analytical PDE solutions. As a result, KAN is not just a problem-solving framework but also a valuable tool for theoretical insights and educational purposes.

Recently, some PINN architectures based on KAN have been proposed, demonstrating significant potential. Wang et al. (2024c) proposed Kolmogorov–Arnold-Informed Neural Network (KINN). Through systematic numerical experiments, it demonstrates KINN's superior performance over MLP-based PINNs in terms of accuracy and convergence speed for a variety of PDEs, including those involving multi-scale, singularity, stress concentration, nonlinear hyperelasticity, and heterogeneous properties. The authors also highlight the potential of KINN in handling complex geometries and propose future research directions, such as incorporating mesh adaptation techniques from finite element methods to enhance KINN's capabilities. It underscores the potential of KANs to revolutionize AI for PDEs. Shukla et al. (2024) introduces a modified version of KANs, known as Chebyshev KANs, which are compared against the conventional MLPs in terms of accuracy, efficiency, and robustness. The study systematically evaluates these models on various benchmarks, including forward and inverse problems in computational physics. The results indicate that while the original KANs based on B-splines struggle with accuracy and efficiency, the modified Chebyshev KANs demonstrate comparable performance to MLPs, albeit with potential divergence issues under certain conditions. The paper further explores the learning dynamics of these models through the lens of the Information Bottleneck theory, revealing insights into their training processes. In Rigas et al. (2024), an adaptive training scheme for Physics-informed Kolmogorov-Arnold networks is proposed, which includes techniques such as residual-based attention (RBA), adaptive resampling of collocation points, and an adaptive state transition method to mitigate the challenges associated with grid updates in KANs. Through comparative experiments on various PDEs, the study demonstrates that PIKANs significantly outperform traditional MLP-based PINNs in terms of accuracy and convergence speed, particularly for problems involving multi-scale dynamics, singularities, and heterogeneous materials.

2.1.6 Transformer

Transformer (Vaswani et al. 2017) architecture represents a significant breakthrough in NLP, replacing RNNs by enabling the parallel processing of input sequences through self-attention mechanisms. This innovation has been essential in addressing long-range dependencies efficiently, overcoming the computational limitations inherent in sequential models. Central to the Transformer is the self-attention module, which allows each position in the sequence to attend to all other positions in the previous layer, facilitating the capture of global context within a single operation. The model follows an encoder–decoder structure, utilizing stacked layers of self-attention and feed-forward networks. The encoder handles the context of the input sequence, while the decoder leverages this context to produce the output sequence. Positional encodings are integrated to convey information about the sequence order. Transformers have played a pivotal role in the development of models such as BERT, GPT, and T5, which have set new standards in tasks like machine translation, text summarization, and question answering. These advancements underscore the transformative influence of the Transformer architecture in reshaping modern NLP.

Santos et al. (2023) introduces the Physical Information Transformer (PIT) model, which is based on Transformer architecture for learning solution operators of PDEs. PIT leverages the attention mechanism to capture the relationship between initial conditions and query points, improving the model’s generalization capabilities. It maintains invariance when handling the discretization of both input and query domains while enabling interaction through cross-attention blocks. Compared to the established physics-based DeepONet (Lu et al. 2021a), PIT shows significant improvements in multiple aspects, demonstrating effective performance in high-dimensional problems even with limited training resources. PIT also offers flexibility in adjusting the number of query and input samples per training instance and discretizing input functions–features not supported by other operator learning models, such as DeepONet. In contrast to operator learning in PINN networks, PIT addresses the issue of poor performance in existing solutions and eliminates the need for assumptions about initial and boundary conditions. The Transformer-based architecture comprises an encoder and decoder. To reduce the automatic partitioning cost associated with PINN loss, PIT avoids self-attention on query and input points. The author also compared PIT with the traditional physics-based DeepONet and an improved model by Wang et al. (2021c). These models were tested on the one-dimensional Burgers equation and the two-dimensional heat equation. The results show that while the computational cost of PIT is similar to DeepONet, its solving performance is significantly improved (Fig. 7).

Additionally, Zhao et al. (2024) proposes PINNsFormer, a Transformer-based architecture that captures temporal dependencies using multi-head attention mechanisms, enhancing the generalization and accuracy of PDE solutions. PINNsFormer transforms point-wise inputs into pseudo sequences, enabling the model to learn temporal dependencies and more effectively propagate initial condition constraints. A key innovation is the introduction of

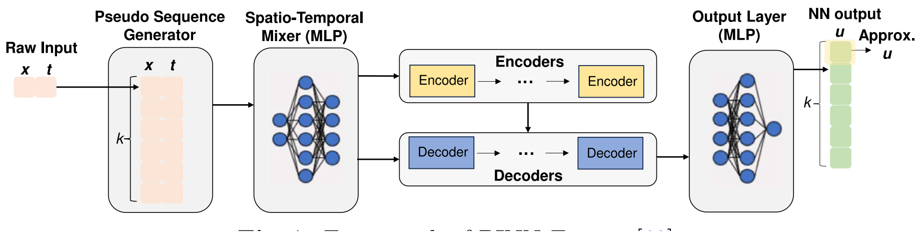

Fig. 7 Framework of PINNsFormer (Zhao et al. 2024)

a new activation function, Wavelet, which anticipates Fourier decomposition in deep neural networks, improving approximation capabilities. The framework also supports various learning schemes, such as Neural Tangent Kernel (NTK), to further enhance performance. Empirical results show that PINNsFormer outperforms conventional PINNs in generalization and accuracy, particularly in scenarios where traditional models fail. It demonstrates significant improvements in prediction accuracy for out-of-distribution data and high-dimensional PDEs. Notably, PINNsFormer reduces the total loss term, indicating more efficient optimization. A loss landscape analysis reveals a smoother landscape with fewer local minima, suggesting easier convergence to the global minimum. Wu et al. (2024) introduces Transolver, a model designed to solve PDEs on complex geometries with high efficiency and accuracy. Leveraging Transformer architecture, Transolver incorporates PhysicsAttention, an adaptive mechanism that segments the discretized domain into learnable slices. These slices group mesh points with similar physical states, enabling the model to capture complex physical correlations with linear computational complexity. The architecture replaces standard attention with PhysicsAttention, processing physics-aware tokens that encode interactions within the discretized domain. Extensive experiments across six benchmarks and large-scale industrial simulations confirm Transolver's consistent state-of-the-art performance, with a 22% relative gain. Its efficiency, scalability, and out-of-distribution generalizability are validated through rigorous testing, demonstrating its potential for real-world applications, including weather forecasting, industrial design, and material analysis.

2.1.7 Other architectures for PINN

In addition to the neural network frameworks discussed above, various novel PINN architectures have been developed (Fig. 8).

Neural architecture search (NAS) has been leveraged to optimize PINNs. For example, Wang et al. (2022c) introduces Auto-PINN, a systematic and automated approach for optimizing the hyperparameters of PINNs. Auto-PINN employs Neural Architecture Search (NAS) techniques to avoid manual or exhaustive searches of the hyperparameter space associated with PINNs. Through a series of pre-experiments using standard PDE benchmarks, they explore the structure-performance relationship in PINNs. They find that different hyperparameters can be decoupled, and the training loss function of PINNs serves as an effective search objective. In Wang and Zhong (2024), a novel framework NAS-PINN is proposed, which also leverages NAS to optimize the design of PINNs. By relaxing the search space into a continuous domain and utilizing masks to enable tensor additions of different shapes, NAS-PINN employs a bi-level optimization process. The inner loop optimizes the weights

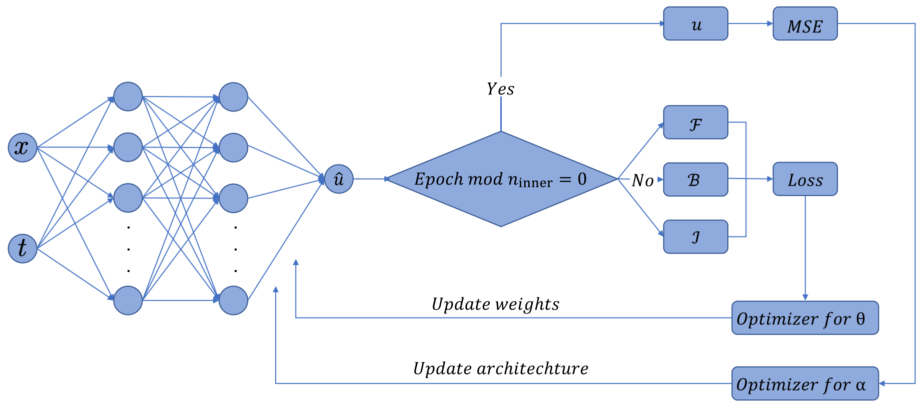

Fig. 8 Framework of PINN with Neural architecture search (NAS-PINN) (Wang and Zhong 2024)

and biases of the neural network, while the outer loop updates the architecture parameters. This approach automates the selection of the optimal neural network architecture, enhancing the efficiency and accuracy of PINNs. Through extensive experiments, it demonstrates that NAS-PINN achieves superior performance compared to manually designed architectures, particularly in handling complex and irregular domains.

Moreover, Cho et al. (2022) introduces Separable PINNs (SPINNs), an innovative architecture aimed at mitigating the curse of dimensionality encountered when solving PDEs. SPINNs leverage forward-mode automatic differentiation to significantly reduce computational overhead, processing inputs on an axis-by-axis basis rather than point-wise. This approach decreases the number of required network forward passes, making computation and memory costs grow more gradually with grid resolution, thereby enhancing scalability. Experimental results demonstrate that SPINNs substantially reduce training run-times while achieving comparable accuracy to standard PINNs across various PDE systems. This advancement positions SPINNs as a promising solution for efficiently tackling high-dimensional PDE problems. Wang et al. (2024b) proposes a novel PINN architecture PirateNets, designed to improve the training stability and efficiency. PirateNets utilize adaptive residual connections, enabling the network to start as a shallow model and progressively deepen during training. This innovative approach mitigates the issues associated with traditional MLPs, which often struggle with derivative trainability and unstable minimization of PDE residual loss. The adaptive initialization scheme allows the network to encode suitable inductive biases specific to the underlying PDE system, thereby improving model performance.

2.1.8 Domain decomposition

Domain Decomposition is a technique used to divide a complex solution domain into multiple smaller subdomains to improve computational efficiency and solution accuracy. Its application in PINNs is particularly suitable for handling large-scale and complex problems. By decomposing the entire solution domain into several subdomains, problems can be solved independently in each subdomain, and parallel computing can speed up the solution process. Interface conditions are introduced between adjacent subdomains to ensure the continuity and consistency of the solution. The domain decomposition methods can flexibly adapt to complex geometries and boundary conditions, making PINNs more versatile and efficient. It is a promising approach for solving complex PDE problems (Fig. 9).

Next, we will introduce some key formulas for domain decomposition methods. The entire solution domain Ω is divided into N subdomains Ω_i,

$$\Omega = \bigcup_{i=1}^{N} \Omega_i \tag{17}$$

The physical equations are solved within each subdomain Ω_i,

$$\mathcal{F}_i(u_i) = 0 \quad \text{in} \quad \Omega_i, \tag{18}$$

where $\mathcal{F}_i$ represents the physical equations in subdomain Ω_i, and u_i is the solution in subdomain Ω_i. The interface conditions between adjacent subdomains are introduced to ensure the continuity and consistency of the solution. Let Γ_{ij} be the interface between subdomains Ω_i and Ω_j, the interface condition is defined as,

$$u_i = u_j \quad \text{on} \quad \Gamma_{ij}. \tag{19}$$

Then the composite loss function is as follows,

$$\mathcal{L} = \sum_{i=1}^{N} \mathcal{L}^{i}_{\text{physics}} + \sum_{i \neq j} \mathcal{L}^{ij}_{\text{interface}}, \tag{20}$$

where the physical loss $\mathcal{L}^{i}_{\text{physics}}$ is computed within each subdomain, and the interface loss $\mathcal{L}^{ij}_{\text{interface}}$ is computed on the interfaces between adjacent subdomains.

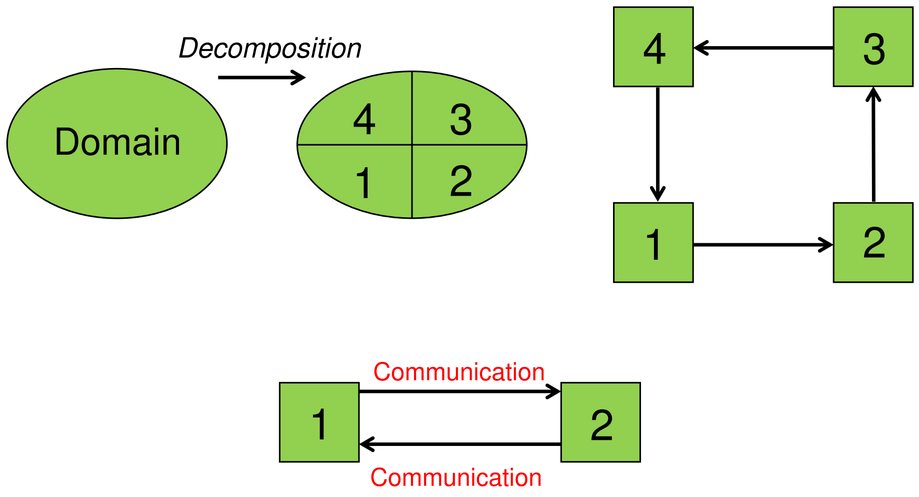

Fig. 9 Domain decomposition methods (Wu et al. 2022) in PINNs

In Karniadakis and Em (2020), Extended PINNs (XPINNs), a generalized space-time domain decomposition approach for solving nonlinear PDEs on complex-geometry domains is proposed. XPINNs can be applied to any type of PDE and allow for arbitrary domain decomposition in both space and time, facilitating efficient parallel computation. Within each subdomain, a separate neural network is deployed with optimally selected hyperparameters. This enables the use of deeper networks in subdomains with complex solutions and shallower networks in simpler regions. Jagtap et al. (2020a) proposed Conservative PINNs (cPINNs) on discrete domains for solving nonlinear conservation laws. It divides the computational domain into discrete subdomains and applies PINNs within each subdomain while enforcing flux continuity across interfaces to ensure global conservation. The cPINN framework provides additional flexibility in choosing optimization algorithms and training parameters, such as the number and placement of residual points, activation functions, and network widths and depths. Similarly, Wu et al. (2022) proposed an improved deep neural network method based on domain decomposition. It can address the limitations of PINNs in handling large-scale complex problems due to the use of a single neural network and gradient pathology.

2.1.9 Activation functions for PINN

Nonlinear activation functions are essential in PINNs as they enable the network to learn and represent the complex, nonlinear relationships inherent in PDEs. Common choices include the hyperbolic tangent (tanh), sigmoid. More recent studies have explored specialized activations such as sinusoidal (sin) functions.

Various novel activation functions have been developed for PINNs. For example, Jagtap et al. (2020b) introduces an adaptive activation function with a scalable hyper-parameter for PINNs, which is dynamically adjusted during training, thereby optimizing the network's topology and significantly improving convergence rates and solution accuracy. The empirical results demonstrate that the adaptive activation function outperforms traditional fixed activation functions, especially in early training phases. The paper also provides a theoretical proof that gradient descent algorithms utilizing the proposed method are not attracted to suboptimal critical points or local minima. Fu et al. (2024) introduces Physics-Informed Kernel Function Neural Networks (PIKFNNs), designed to solve a variety of linear and specific nonlinear PDEs. PIKFNNs employ a one-hidden-layer shallow neural network structure with physics-informed kernel functions (PIKFs) as custom activation functions. These PIKFs incorporate PDE information, such as fundamental solutions, Green's functions, harmonic functions, and solutions to simplified linear PDEs, enhancing the network's ability to accurately approximate complex solutions. It demonstrates that PIKFNNs offer significant improvements in accuracy and efficiency over traditional PINNs, particularly in handling irregular domains and complex geometries. Abbasi and Andersen (2024) proposes Physical Activation Functions (PAFs), which are derived from the mathematical expressions inherent in the physical systems under study, rather than relying solely on standard activation functions like tanh or sigmoid. PAFs can be integrated into neural networks in an explicit or self-adaptive manner, with the latter allowing for automatic determination of the relative impact of PAFs for each neuron. The study demonstrates that PAFs significantly improve the accuracy of PINN predictions, particularly for out-of-training data, and reduce the network size by up to 75% while maintaining accuracy.

Moreover, Zeng et al. (2022) introduces innovative adaptive techniques to enhance the computational performance of deep neural networks (DNNs) in solving high-dimensional PDEs. The authors propose three key strategies: adaptive loss functions, adaptive activation functions, and adaptive sampling. These methods are designed to improve the accuracy and convergence speed of DNNs without increasing the network complexity. The adaptive loss function modifies the traditional L^2 norm, introducing a parameter α that transforms the loss function based on the training phase. The adaptive activation function varies the activation across layers, and the adaptive sampling strategy allocates more points to regions with higher local residual errors. Numerical experiments demonstrate that these techniques significantly outperform conventional methods, achieving relative errors on the order of $O(10^{-4})$ for some 50-dimensional problems. The study provides a promising adaptive approach to enhance DNN performance in computational physics and engineering.

2.2 Adaptive data resampling for PINN

The performance of PINNs heavily relies on the strategic selection of sampling points to effectively train the network. The location or distribution of these points is crucial for accurately approximating the solutions.

In most PINNs, the collection points for training (usually called residual points) are specified at the beginning of training and not changed during the training process. Various sampling methods can used in PINNs, such as uniform sampling, random sampling, Latin hypercube sampling (LHS) (Stein 1987; Mckay et al. 2000), Halton sequence (Halton 1960; Berblinger and Schlier 1991), Hammersley sequence (Wong et al. 1997), Sobel sequence (Sobol' 1967). To increase the accuracy of PINN, an alternative way is to select a new set of residual points for training in every certain iteration (Lu et al. 2021b). Even different sampling methods can be used at different times.

While fixed sampling strategies are widely used due to their simplicity, they may not always be the most efficient or accurate method, especially for problems with complex geometries or solutions that exhibit sharp gradients. More recently, adaptive sampling strategies have been developed to dynamically adjust the distribution of sampling points based on the current state of the network's training. These adaptive methods have shown significant improvements in the accuracy of PINNs with fewer residual points. Lu et al. (2021b) proposed the first adaptive sampling for PINNs, the residual-based adaptive refinement (RAR) method, which adds new residual points in the locations with large PDE residuals. In Nabian et al. (2021), an importance sampling approach to enhance the training efficiency of PINNs is introduced. By sampling collocation points according to a distribution proportional to the loss function, the method accelerates convergence and improves training effectiveness. The technique is straightforward to implement, does not introduce new hyperparameters, and is demonstrated through numerical examples to significantly enhance computational efficiency. Wu et al. (2023) provides a comprehensive study of sampling strategies for PINNs, examining both non-adaptive and adaptive resampling methods. It proposed two new adaptive nonuniform sampling techniques: residual-based adaptive distribution (RAD) and residual-based adaptive refinement with distribution (RAR-D). These methods dynamically adjust the distribution of residual points according to a designed probability density function (PDF) to enhance the training efficiency and accuracy of PINNs. In Daw et al. (2023), a novel Retain-Resample-Release (R3) sampling algorithm is proposed to address the issue of propagation failures in PINNs. It hypothesizes that successful training of PINNs depends on the effective propagation of solutions from initial/boundary conditions to interior points. Propagation failures, characterized by imbalanced PDE residual fields, can cause PINNs to converge to trivial solutions. The R3 algorithm incrementally accumulates collocation points in regions of high PDE residuals, mitigating these failures with minimal computational overhead.

Furthermore, Tang et al. (2023) proposed DAS-PINNs, which use deep generative models to dynamically refine the training set by generating new collocation points. Specifically, the method treats the residual as a probability density function and uses a generative model, termed KRnet, to sample points consistent with this distribution–more samples are placed in regions of higher residual. Yang et al. (2023) proposed dynamic mesh-based importance sampling (DMIS) to address the computational inefficiency and unstable convergence issues inherent in PINN training. DMIS integrates importance sampling into the training process by constructing a dynamic triangular mesh that efficiently estimates sample weights. Evaluations on three benchmarks–Schrödinger Equation, Viscous Burgers' Equation, and Korteweg-de Vries Equation–demonstrate DMIS's superior performance in improving convergence speed and accuracy. In Jiao et al. (2024), a Gaussian Mixture Distribution-based Adaptive Sampling (GAS) method is proposed for PINNs. Inspired by incremental learning, the authors propose a risk min–max framework that dynamically refines the sampling of collocation points. GAS uses the current residual information to generate a Gaussian mixture distribution for sampling additional points, accelerating convergence and improving accuracy. Lau et al. (2024) introduced PINNACLE (PINN Adaptive Collocation and Experimental Points Selection), an algorithm designed to optimize the selection of all types of training points for PINNs. Unlike previous approaches that focus solely on collocation or experimental points, PINNACLE jointly optimizes the selection of both, adjusting the proportion of collocation point types as training progresses. Tang et al. (2024) proposed Adversarial Adaptive Sampling (AAS), a novel method that integrates PINNs with optimal transport theory. AAS employs a deep generative model to adjust the distribution of random samples in the training set, ensuring the residual induced by the neural network maintains a smooth profile. By embedding the Wasserstein distance between the residual-induced distribution and a uniform distribution into the loss function, AAS minimizes the statistical errors introduced by random samples. All these methods enhance the efficiency of network training, thereby significantly enhancing the performance of PINNs in various PDE problems.

Inspired by the above research, we have proposed an enhanced hybrid adaptive PINN (Luo et al. 2025). It employs a hybrid adaptive (HA) sampling method which ensures randomness in sampling points while also giving due consideration to points with large PDE residuals during the training procedure. Two distinct resampling strategies are incorporated in HA method during training. Some examples are present in Table 3 and Fig. 10, which have demonstrated our proposed method outperforms state-of-the-art resampling methods.

2.3 Loss function and optimization for PINN

In PINNs, loss functions play pivotal roles in model training and performance. The loss function impacts convergence and the network's ability to learn effectively.

Table 3 Example of 1D Poisson equation

Methods		Epochs = 20000		Epochs = 40000	
		$N_r = 14$	$N_r = 26$	$N_r = 14$	$N_r = 26$
PINN		$18.07\% \pm 3.70\%$	$6.82\% \pm 4.11\%$	$48.28\% \pm 36.22\%$	$15.30\% \pm 13.95\%$
PINN Random-R		$3.76\% \pm 2.02\%$	$\underline{0.99\% \pm 1.37\%}$	$2.52\% \pm 1.89\%$	$\underline{0.14\% \pm 0.06\%}$
RAD (Wu et al. 2023)	k = 1, c = 1	$5.49\% \pm 4.53\%$	$2.23\% \pm 2.54\%$	$\underline{1.62\% \pm 1.28\%}$	$1.02\% \pm 1.53\%$
	k = 1, c = 2	$\underline{3.31\% \pm 1.57\%}$	$2.51\% \pm 2.81\%$	$2.21\% \pm 1.40\%$	$0.82\% \pm 0.82\%$
	k = 2, c = 1	$5.44\% \pm 4.95\%$	$3.78\% \pm 2.30\%$	$2.04\% \pm 2.06\%$	$1.11\% \pm 0.65\%$
	k = 2, c = 2	$6.00\% \pm 5.49\%$	$2.50\% \pm 2.44\%$	$1.87\% \pm 1.59\%$	$0.89\% \pm 1.01\%$
PINN HA (ours)		$\mathbf{2.73\% \pm 2.38\%}$	$\mathbf{0.24\% \pm 0.21\%}$	$\mathbf{1.28\% \pm 0.76\%}$	$\mathbf{0.11\% \pm 0.08\%}$

L2 relative errors of comparative methods. The experiment is repeated ten times, and the mean and standard deviation of L2 relative errors are computed. N_r denotes the number of residual points

2.3.1 Loss reweighting

In PINNs, the PDE loss, initial and boundary condition losses are typically added together and optimized directly. However, the scale and convergence speed of different loss terms might be different. The imbalances among these losses can lead to deteriorated performance. To mitigate these issues, various loss reweighting methods have been proposed, which adaptively adjust the relative importance of different components in the loss function.

Wang et al. (2021c) examine gradient flow pathologies in PINNs and propose solutions to enhance their predictive accuracy. It introduces a learning rate annealing algorithm that leverages gradient statistics to balance the composite loss functions and proposes a novel neural network architecture more resilient to gradient issues. These developments improve the accuracy of PINNs by factors of 50–100 times across a range of computational physics problems. It provides new insights into training constrained neural networks and offers practical methods to mitigate gradient flow pathologies. Wang et al. (2022a) investigates the training dynamics of PINNs using the Neural Tangent Kernel (NTK) framework. It derives the NTK for PINNs and proves that, under certain conditions, it converges to a deterministic kernel that remains constant during training in the infinite-width limit. This allows the analysis of PINN training dynamics through the lens of NTK, revealing discrepancies in the convergence rates of different loss components. To address these issues, it proposes a novel gradient descent algorithm that utilizes the eigenvalues of the NTK to balance the training process, thereby improving the stability and effectiveness of PINNs. In Xiang et al. (2022), a self-adaptive loss balanced PINNs (lbPINNs) is proposed, which uses a self-adaptive mechanism that dynamically adjusts the weights of the different loss terms during training. This is achieved through probabilistic modeling and maximum likelihood estimation. It establishes a Gaussian probabilistic model with output u. The Gaussian likelihood is defined as Gaussian with mean given by the approximation of PINNs $\hat{u}(x,t;\theta)$ and the uncertainty parameter ε_d,

$$p(u|\hat{u}(x,t;\theta)) = N\left(\hat{u}(x,t;\theta), \varepsilon_d^2\right) \tag{21}$$

The uncertainty parameters are adjusted using maximum likelihood inference. To achieve this, the negative log-likelihood of the model is minimized,

$$\begin{aligned} -\log p(\boldsymbol{u}|\hat{\boldsymbol{u}}(\boldsymbol{x},t;\theta)) &\propto \tfrac{1}{2\varepsilon_d^2}\left\|\boldsymbol{u}-\hat{\boldsymbol{u}}(\boldsymbol{x},t;\theta)\right\|^2 + \log\varepsilon_d \\ &= \tfrac{1}{2\varepsilon_d^2} L_{data}(\theta) + \log\varepsilon_d. \end{aligned} \tag{22}$$

Furthermore, assume that the output of our Gaussian probability model includes two vectors u, g, each following a Gaussian distribution,

$$\begin{aligned} p(u,g|\hat{u}(x,t;\theta)) &= p(u|\hat{u}(x,t;\theta)) \cdot p(g|\hat{u}(x,t;\theta)) \\ &= N\left(\hat{u}(x,t;\theta), \varepsilon_d^2\right) \cdot N\left(\hat{u}(x,t;\theta), \varepsilon_b^2\right). \end{aligned} \tag{23}$$

Therefore, it establishes the multi-output model with four vectors to define the complex loss function. The loss function of lbPINNs can be formulated as,

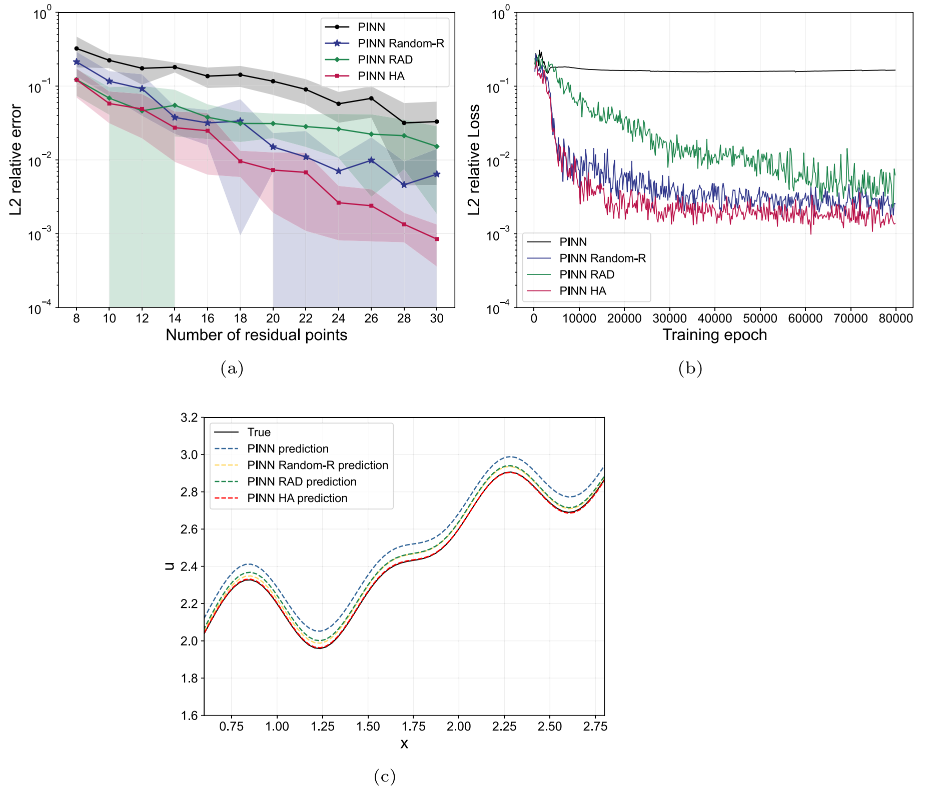

Fig. 10 The performance of our proposed HA sampling method for 1D Poisson equation

$$L(\epsilon;\theta;N) = \frac{1}{2\epsilon_f^2} L_{PDE}\big(\theta;N_f\big) + \frac{1}{2\epsilon_b^2} L_{BC}(\theta;N_b) + \frac{1}{2\epsilon_i^2} L_{IC}(\theta;N_i) + \frac{1}{2\epsilon_d^2} L_{data}(\theta;N_{data}) + \log \epsilon_f \epsilon_b \epsilon_i \epsilon_d \epsilon_d, \tag{24}$$

where $\epsilon = \{\epsilon_f, \epsilon_b, \epsilon_i, \epsilon_d\}$ describes the adaptive weights of loss terms. $\omega = \{\omega_f, \omega_b, \omega_i, \omega_d\}$, $\omega := \frac{1}{2\varepsilon^2}$ describes the total weights of loss terms. Although lbPINNs have made progress in designing fast and accurate scientific machine learning techniques, some problems need to be resolved, such as (1) It is a common phenomenon that the adaptive weight of PDE loss decreases slowly in the above experiment. Therefore, we would conduct a theoretical analysis of this phenomenon in order to improve the robustness and scalability of the method. (2) It is worth exploring whether this probabilistic model is the most appropriate one, and under certain assumptions, it is possible to establish a unified framework with several other loss balancing algorithms. Moreover, different weights for each loss term can be even adjusted at each training point. For example, Gu et al. (2021) proposes SelectNet, a novel self-paced learning framework designed to improve the convergence of PINNs. By incorporating a selection network that adaptively weights training samples, SelectNet prioritizes easier samples in the early stages of training and gradually incorporates more challenging ones.

Further, McClenny and Braga-Neto (2023) introduces self-adaptive PINNs (SA-PINNs), which employ fully trainable adaptation weights that apply individual attention to each training point, enabling the network to autonomously identify and focus on challenging regions of the solution. The adaptive weights create a soft attention mask, increasing as the corresponding losses increase. It also demonstrates how to construct a continuous map of adaptive weights using Gaussian process regression, facilitating the use of stochastic gradient descent in scenarios where conventional gradient descent is insufficient. Experiments show that SA-PINNs improve the accuracy and robustness of PDE solutions. Song et al. (2024) introduces a novel architecture named loss-attentional PINN (LA-PINN) to tackle the issue that the vanilla PINN suffers from slow convergence speed and poor accuracy at hard-to-fit regions. This architecture pays attention to the control of loss error at each training point by allocating every independent loss component with a loss-attentional net (LAN). Proceeding by feeding point squared error (SE) into LAN, the "attentional function" could be built to accomplish the task of distributing different weights to diverse point errors. The LA-PINN architecture employs a point error-based weighting method that utilizes adversarial training between the main network and the LANs. The main network, responsible for predicting the solution, minimizes the loss through gradient descent, while the LANs adjust the weights of the SEs through gradient ascent. This mechanism ensures that more attention is given to hard-to-fit points by increasing the growth rates of both the weights and the update gradients for the SEs at these points. Through this dynamic weighting approach, LA-PINN demonstrates superior predictive performance and faster convergence compared to vanilla PINNs and other state-of-the-art methods. Numerical experiments confirm the efficacy of LA-PINN in advancing the convergence of hard-to-fit points and achieving high accuracy in solving PDEs.

2.3.2 Novel loss functions

In addition to the loss reweighting methods discussed above, numerous variant loss functions have been also developed.

Regularization is an important technique for enhancing the training and generalization capabilities of machine learning models. Some research has proposed various regularization methods for PINN loss functions. Yu et al. (2022) proposes gradient-enhanced PINNs (gPINNs), which integrate gradient information into the neural network's loss function, enhancing the model's performance. In traditional PINNs, the primary goal is to constrain the PDE residual f to zero, ensuring that at each point $\boldsymbol{x}$, $f(\boldsymbol{x}) = 0$. However, since the residual $f(\boldsymbol{x})$ is zero at any point $\boldsymbol{x}$, it can be inferred that its derivative should also be zero. Thus, gPINNs assume that the exact solution of the PDE is sufficiently smooth so that the gradient of the PDE residual $\nabla f(\boldsymbol{x})$ exists and propose enhancing PINNs by enforcing that the derivatives of the PDE residual are also zero. Then the loss function of gPINNs is given by,

$$\mathcal{L} = w_f \mathcal{L}_f + w_b \mathcal{L}_b + w_i \mathcal{L}_i + \sum_{i=1}^{d} w_{g_i} \mathcal{L}_{g_i}\left(\theta; \mathcal{T}_{g_i}\right), \tag{25}$$

where

$$\mathcal{L}_f(\theta; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathrm{x} \in \mathcal{T}_f} \left| f\left(\mathrm{x}; \frac{\partial \hat{u}}{\partial x_1}, \ldots, \frac{\partial \hat{u}}{\partial x_d}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_d}; \ldots; \lambda\right)\right|^2, \tag{26}$$

$$\mathcal{L}_b(\theta; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathrm{x} \in \mathcal{T}_b} |\mathcal{B}(\hat{u}, \mathrm{x})|^2, \tag{27}$$

$$\mathcal{L}_{g_i}(\theta; \mathcal{T}_{g_i}) = \frac{1}{|\mathcal{T}_{g_i}|} \sum_{\mathrm{x} \in \mathcal{T}_{g_i}} \left|\frac{\partial f}{\partial x_i}\right|^2, \tag{28}$$

$$\mathcal{L}_i(\theta, \lambda; \mathcal{T}_i) = \frac{1}{|\mathcal{T}_i|} \sum_{\mathrm{x} \in \mathcal{T}_i} |\hat{u}(\mathrm{x}) - u(\mathrm{x})|^2, \tag{29}$$

where w_f, w_b, w_i and w_g are the weights. The advantages of gPINNs include improved accuracy, efficiency, and better generalization to regions without training points. They often require fewer training points to achieve similar or better accuracy than traditional PINNs, making them more computationally efficient. By leveraging gradient information, gPINNs generalize better, resulting in more robust solutions. When combined with RAR, gPINNs perform exceptionally well, particularly for PDEs with solutions that have steep gradients. This makes gPINNs a powerful tool for solving both forward and inverse PDE problems in various fields, such as optics, fluid mechanics, systems biology, and biomedicine. Overall, gPINNs represent a significant advancement in the application of deep learning to PDEs, offering enhanced performance and efficiency. In Wang et al. (2024a), a practical PINN framework is specifically designed to address multi-scale problems with multi-magnitude loss terms. The proposed method enhances the conventional PINN approach by implementing a grouping regularization strategy, which normalizes the scale of different loss components. This ensures that each term contributes equally to the overall loss function, thereby improving the network's ability to capture both macroscopic and microscopic features of the solution. Additionally, the framework incorporates specialized neural network architectures, such as Fourier feature networks, to further enhance accuracy and efficiency. Through numerical experiments, it demonstrates that this improved PINN framework significantly outperforms standard PINNs in solving multi-scale PDEs, making it a valuable tool for applications in complex, multi-scale phenomena.

Moreover, Wang et al. (2022b) challenges the conventional use of L^2 loss in training PINNs for solving PDEs. It scrutinizes the relationship between the loss function and the accuracy of the learned solution, introducing the concept of stability from PDE theory to analyze the asymptotic behavior of solutions as the loss approaches zero. Focusing on the Hamilton–Jacobi–Bellman (HJB) equations, pivotal in optimal control, the study reveals that for general L^p loss functions, stability is achieved only with sufficiently large p. This finding suggests that the ubiquitous L^2 loss may be suboptimal for HJB equations, advocating for the use of L^∞ loss. The paper presents a novel PINN training algorithm that employs a min-max optimization strategy akin to adversarial training to minimize the L^∞ loss, demonstrating improved solution accuracy through empirical experiments. The research provides critical insights into loss function design for PINNs, particularly for high-dimensional nonlinear PDEs. Wang et al. (2024d) investigates the impact of different loss functions on the training of PINNs. It categorizes loss functions into two types: observation data loss, which directly constrains and measures the model output, and model loss, which includes information from governing equations and variational forms, influencing the network's performance indirectly. It uncovers a stable "loss-jump" phenomenon: when transitioning from data loss to model loss, which encompasses derivative information of varying orders, the neural network solution experiences a significant deviation from the exact solution. Further experiments reveal that this phenomenon arises from the different frequency preferences of neural networks under various loss functions. Theoretical analysis elucidates the frequency preference under model loss, providing insights into the underlying mechanisms of PINNs in solving PDEs. This work offers a valuable perspective for optimizing the training process and enhancing the accuracy of PINNs.

2.4 Feature embedding and augmentation

Feature embedding is widely utilized across various domains of machine learning, such as Natural Language Processing (NLP) (Patil et al. 2023), Computer Vision (CV) (Kan et al. 2019), and Recommendation Systems (Zhang et al. 2016). The feature (or variable) of interest is mapped into a a vector space, where relationships and patterns can be more easily discerned by algorithms. NeRF (Mildenhall et al. 2020) is a typical example, spatial encoding efficiently represents and processes information regarding the positions of input points. This encoding integrates the spatial position of each input point with additional pertinent details like color and radiation levels, enhancing the quality of image reconstruction and rendering. NeRF employs sine and cosine functions to encode coordinates in three-dimensional space as follows,

$$\begin{aligned}\gamma(x,y,z) = \Big(&x, y, z, \sin(2^0, x), \sin(2^0, y), \sin(2^0, z), \cos(2^0, x), \cos(2^0, y), \cos(2^0, z),\\ &\sin(2^1, x), \sin(2^1, y), \sin(2^1, z), \cos(2^1, x), \cos(2^1, y), \cos(2^1, z), \dots,\\ &\sin(2^{n-1}, x), \sin(2^{n-1}, y), \sin(2^{n-1}, z), \cos(2^{n-1}, x), \cos(2^{n-1}, y), \cos(2^{n-1}, z)\Big)\end{aligned} \tag{30}$$

Similarly, Tancik et al. (2020) proves that Fourier feature mapping can effectively mitigate the spectral bias of coordinate-based MLPs towards low frequencies, based on the Neural Tangent Kernel (NTK) theory (Jacot et al. 2018). A randomly selected Fourier feature mapping with a suitable scale is employed,

$$\gamma(\mathrm{v}) = \left(a_1 \cos(2\pi \mathrm{b}_1^T \mathrm{v}), a_1 \sin(2\pi \mathrm{b}_1^T \mathrm{v}), \dots, a_m \cos(2\pi \mathrm{b}_m^T \mathrm{v}), a_m \sin(2\pi \mathrm{b}_m^T \mathrm{v})\right)^T. \tag{31}$$

Experiments demonstrate that this approach improves the performance of coordinate-based MLPs across various low-dimensional tasks in computer vision and graphics (Fig. 11).

Inspired by the above works, numerous novel PINNs incorporating feature embedding have been proposed to improve the performance. A fundamental weakness of PINNs is spectral bias (Rahaman et al. 2019), a recognized challenge in fully-connected neural networks that restricts their capacity to effectively learn high-frequency functions. In Wang et al. (2021a), a new PINN architecture is introduced that utilizes such spatio-temporal and multi-scale random Fourier features. It elucidates how these coordinate embedding

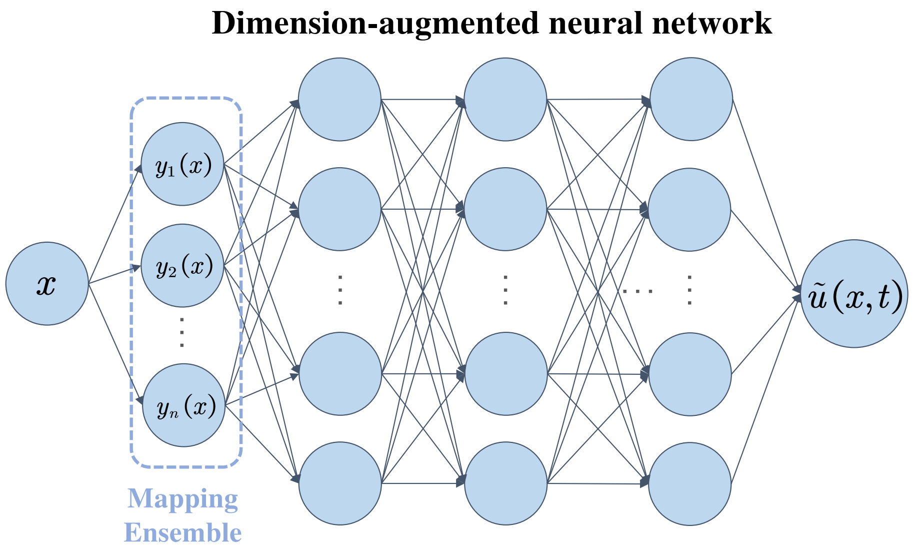

Fig. 11 The framework of dimension-augmented PINN (DaPINN) (Guan et al. 2023)

layers enhance the robustness and accuracy of PINN models, effectively addressing high-frequency and multi-scale challenges. The similar Fourier feature mapping method is also utilized in Li et al. (2024a, c). Peng et al. (2020) proposed a Prior Dictionary PINN (PD-PINN) equipped with task-dependent dictionaries. The PD-PINN demonstrates enhanced representational power in tasks by effectively capturing features from dictionaries, thereby achieving faster convergence during training. Additionally, Wong et al. (2024) proposed the sf-PINN, where the sinusoidal mapping of inputs is utilized. This approach can effectively increase input gradient variability, thereby avoiding trapping in deceptive local minima. The model accuracy can be improved by a few orders of magnitudes with sf-PINN.

Moreover, some augmentation techniques are commonly employed in some CV tasks (Perez and Wang 2017), including spatial transformations (flip, rotate, crop, etc.), noise injection, color transformations, and others. Similarly, more features can be inserted into the input vector of PINN to improve the model's capability. We have proposed a dimension-augmented PINN (DaPINN) (Guan et al. 2023) that systematically manipulates the input dimensionality of the network. The performance of DaPINN have been evaluated in various forward and inverse PDEs, including Poisson equation, Heat equation, Diffusion problems, Burgers equation, and so on. The DaPINN model enhances the solution accuracy through replica augmentation, power series augmentation, and Fourier series augmentation. For example, consider the following 1D Poisson equation,

$$-\Delta u = \sum_{k=1}^{3} k \sin(kx) + 7\sin(7x) + 8\sin(8x)\,,\ \ x \in [0,\pi]\,, \tag{32}$$

with the Dirichlet boundary conditions $u(x=0)=0$ and $u(x=\pi)=\pi$. Figure 12 shows the performance of our proposed DaPINN for 1D Poisson equation and comparison with PINN. Another example for Burgers equation result is shown in Fig. 13.

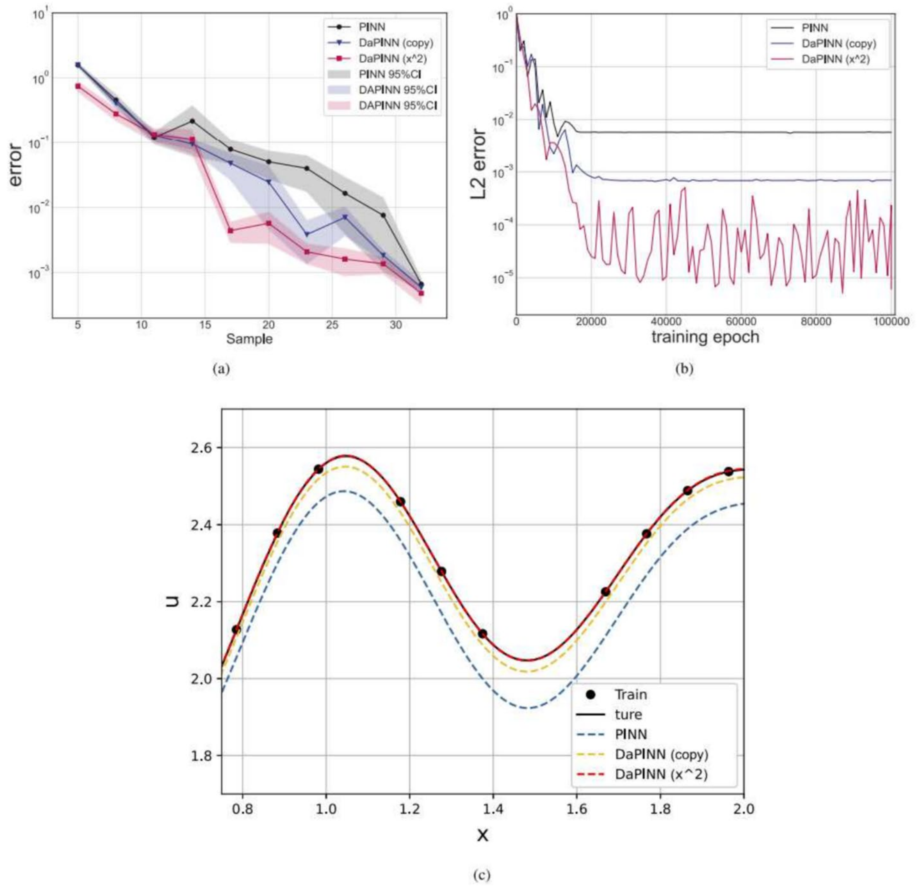

Fig. 12 The performance of DaPINN (Guan et al. 2023) for 1D Poisson equation

The DaPINN models outperform the PINN by expanding the network input dimension, which allows the neural network to extract more informative features.

3 Applications

Recent advancements in the field have demonstrated the effectiveness of PINNs across a broad spectrum of applications. It marks a significant leap forward in the intersection of machine learning and computational physics. Below, we will discuss several key areas where PINNs are making an impact.

3.1 Fluid dynamics

PINNs have emerged as a pivotal technique in fluid mechanics, offering a novel approach to solving the Navier–Stokes equations (NSE) that govern fluid motion (Cai et al. 2021; Mao et al. 2020). The integration of physics and data is particularly powerful for addressing com-

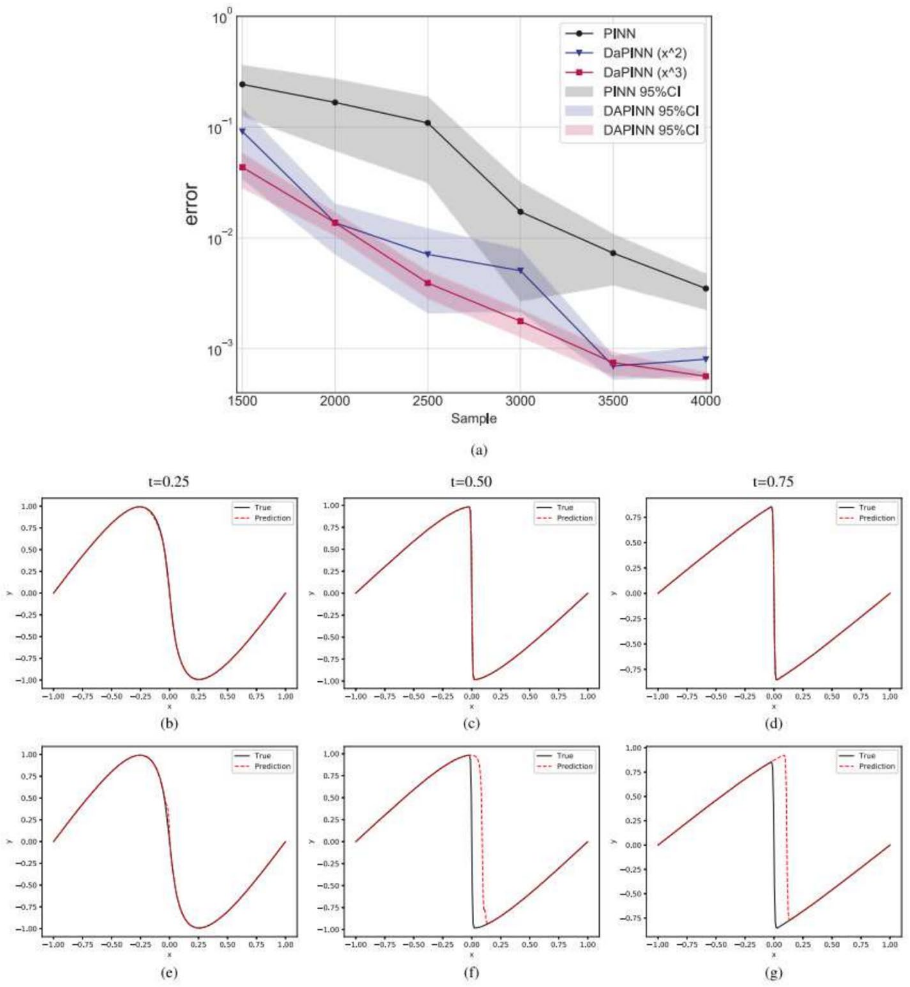

Fig. 13 The performance of DaPINN (Guan et al. 2023) for Burgers equation

plex, high-dimensional fluid flow problems, where traditional computational fluid dynamics (CFD) methods may struggle with the incorporation of noisy data or the requirement for efficient handling of inverse problems.

PINNs have been successfully applied to model three-dimensional wake flows behind bluff bodies, such as cylinders, where they can reconstruct the full velocity and pressure fields from limited observational data. This capability is particularly valuable for diagnostic purposes in complex flows where only partial measurements are available, such as in planar particle image velocimetry (PIV) experiments. Moreover, PINNs are instrumental in simulating high-speed compressible flows, such as those involving shock waves. In these scenarios, traditional CFD methods often rely on precise boundary conditions, which may not be readily available in experimental settings. PINNs, however, can leverage available data, such as density gradients and surface pressure measurements, to infer the complete flow field. This approach has been shown to be effective even in the absence of full boundary condition information, highlighting the flexibility of PINNs in addressing ill-posed problems. The application of PINNs extends to biomedical fluid mechanics as well. For example, in hemodynamics (Kissas et al. 2020), PINNs have been used to simulate blood flow dynamics, estimate wall shear stresses, and predict pulse wave propagation in patient-specific arterial networks.

3.2 Solid mechanics

In solid mechanics, PINNs have been effectively applied to model linear elasticity, elasto-plasticity, hyperelasticity, and fracture mechanics. PINNs can handle forward problems, predicting material responses given inputs. For example, Rao et al. (2021) solved the linear elastic dynamics problem by using PINNs strong form algorithm. PINNs exhibit superior accuracy and efficiency compared to Finite difference methods (FDM) and Finite element methods (FEM) when addressing highly non-uniform forward problems (Guo et al. 2022). The inverse problems of solid mechanics can also be solved, which mainly includes the constitutive equation (Haghighat et al. 2021b) and geometric topology (Zhang et al. 2022a).

The potential of PINNs in solid mechanics is vast, promising to enhance our understanding and prediction of material behavior, leading to innovative solutions in structural design and material science.

3.3 Electromagnetics and optics

PINNs have been extended to solve problems in optics and electromagnetics, such as the 3-D Helmholtz equation and quasi-linear PDE operators. They have proven effective for inverse problems in these domains, which can be particularly challenging due to the intricate nature of electromagnetic fields (Fang and Zhan 2020). Chen et al. (2020a) utilizes PINNs to solve inverse scattering problems in photonic metamaterials and nano-optics technologies. The mesh-free PINNs have been applied to the difficult task of retrieving the effective permittivity parameters of many finite-size scattering systems that involve many interacting nanostructures as well as multi-component nanoparticles.

PINNs represent a transformative approach to tackling complex problems across various fields, such as fluid dynamics, solid mechanics, electromagnetics and optics. Their versatility and efficiency continue to drive research and applications, positioning them as a key component of future computational modeling efforts.

4 Software

A number of open-source tools and libraries have been developed to facilitate the implementation of PINNs. These tools abstract much of the complexity involved in building PINNs, making it easier for researchers and practitioners to apply this methodology to various problems in science and engineering. This section provides an overview of some widely used frameworks and libraries that are publicly available for implementing PINNs (Table 4).

Table 4 Some open-source frameworks for PINN

Framework names	Backend
DeepXDE	TensorFlow, PyTorch, JAX
IDRLnet	PyTorch
NeuroDiffEq	PyTorch
SciANN	TensorFlow
TensorDiffEq	Tensorflow

4.1 DeepXDE

DeepXDE (Lu et al. 2021b) is an open-source Python library specifically designed for solving differential equations using PINNs. It provides a high-level, easy-to-use interface for defining, training, and solving both ODEs and PDEs by leveraging deep learning techniques. The library integrates seamlessly with TensorFlow or PyTorch, and it allows users to easily apply neural networks to complex, real-world physics-based problems.

4.2 IDRLnet

IDRLnet (Peng et al. 2021) is a Python toolbox for systematically modeling and solving problems through PINN. It provides a structured way to integrate geometric objects, data sources, artificial neural networks, loss metrics, and optimizers within Python.

4.3 NeuroDiffEq

NeuroDiffEq (Chen et al. 2020b; Liu et al. 2025) is an open-source Python library developed for solving differential equations using neural networks, particularly within the PyTorch framework. It leverages the power of artificial neural networks to approximate solutions for both ordinary differential equations (ODEs) and partial differential equations (PDEs), subject to specified initial or boundary conditions. Designed for flexibility, the library enables researchers and practitioners to address a wide range of user-defined problems. Furthermore, it facilitates the computation of solutions that are not only continuous but also differentiable, offering a significant advantage over traditional numerical methods such as finite difference and finite element methods.

4.4 SciANN

SciANN (Haghighat and Juanes 2021a) is a high-level neural network API designed specifically for scientific computing. It is built on top of Keras and TensorFlow, making it easy for scientists to leverage deep learning for physics-informed modeling. SciANN allows researchers to define inputs, outputs, and constraints, enabling the formulation of scientific problems as deep learning tasks. The framework is particularly useful for solving PDEs and simulating system dynamics, making it a valuable tool for scientists and engineers looking to apply deep learning techniques to their research.

4.5 TensorDiffEq

TensorDiffEq (McClenny et al. 2021) is a Python package constructed on top of TensorFlow, designed to offer scalable and efficient solvers for PINNs. It is particularly focused on the scalable solving of PINNs for both inference and discovery of inverse problems. What distinguishes TensorDiffEq is its full support for Self-Adaptive PINN solvers and its status as the only fully open-source multi-GPU PINN solution suite.

5 Challenges and outlooks

PINNs have proven to be a versatile tool across various scientific and engineering disciplines, offering a robust and flexible approach to solving complex PDEs where traditional methods may struggle. Despite their successes, ongoing research continues to address theoretical issues and practical limitations, aiming to enhance the scalability, interpretability, and reliability of PINNs. In the following section, we highlight the key challenges associated with PINNs and propose potential directions for future research.

5.1 Challenges for PINN

PINN's ability to address problems in various domains such as fluid dynamics, materials science, and biomechanics has demonstrated the potential of this method to bridge the gap between data-driven models and traditional numerical simulations. However, despite their impressive capabilities, PINNs still face several limitations and challenges that hinder their widespread adoption. PINN consists of three key elements: (1) the neural networks that approximate the underlying function, (2) the physics-informed loss functions that integrate physical laws, and (3) the optimization methods used to train the model. These elements enable PINNs to effectively embed physical constraints into the learning process. However, the complexity of managing these components in practice often leads to issues such as slow convergence, sensitivity to hyperparameters, and difficulty in scaling to high-dimensional or multi-physics problems.

In order to improve the current convergence accuracy of the PINNs, which is around 10^{-3}, rigorous analysis is needed to evaluate PINN's learning abilities, focusing on aspects like generalization, robustness, and the limitations imposed by the complexity of the physical systems. Some studies offer theoretical insights into PINNs (Krishnapriyan et al. 2021; Wang et al. 2022a; Hu et al. 2022), but there's still a need for deeper research on this subject. For instance, Krishnapriyan et al. (2021) identifies a critical "gradient pathologies" issue that arises due to numerical stiffness, leading to imbalanced gradients during backpropagation. This imbalance impedes the network's ability to accurately satisfy both the PDE constraints and boundary conditions, often resulting in suboptimal solutions.

One of the key issues for PINN is the difficulty in capturing fast-varying or high-frequency solutions. Traditional optimization algorithms, such as Adam or L-BFGS, may struggle to converge efficiently in these scenarios, leading to poor performance or slow training. Additionally, multi-physics problems, where different physical processes interact, introduce an extra layer of complexity that current optimization techniques may not handle well. To address these challenges, the development of new optimization algorithms tailored for PINNs is essential. These algorithms should enhance the robustness and efficiency of the training process by adapting to the specific nature of the problem. Potential approaches include designing more sophisticated loss functions, incorporating hierarchical models, or leveraging advanced optimization strategies. These advancements are crucial to improve the scalability, accuracy, and reliability of PINNs, making them more applicable to real-world problems involving high-frequency dynamics and multi-physics interactions.

Another challenge is the inability of current PINNs to handle noisy or incomplete data efficiently. While PINNs integrate physical laws into the learning process, they are still vulnerable to errors and inconsistencies in the training data, which can impact the accuracy of predictions. Furthermore, for multi-scale and multi-physics problems, the scalability of PINNs remains a challenge, as the interaction of different physical models often results in increased model complexity.

Additionally, the development of valuable experimental datasets, along with the corresponding physical models and parameters, is still needed. High-quality datasets are essential for training robust and accurate PINN models, but the lack of accessible, detailed experimental data in many scientific and engineering domains limits the potential for real-world applications. Furthermore, the need for accurate and computationally efficient physical models and parameters that can be integrated into PINNs poses another significant hurdle.

5.2 Operator learning

While PINNs have shown remarkable success in solving complex physical problems by incorporating physical laws into the learning process, several challenges remain that hinder their full potential. One of the key limitations of PINNs is their reliance on the manual formulation of physical constraints, which can become cumbersome and challenging for problems with complex or unknown physics. Furthermore, the performance of PINNs is highly sensitive to the choice of neural network architecture, training strategies, and the handling of boundary conditions, which often require substantial tuning.

In recent years, there has been a shift toward deep operator learning, a promising paradigm that extends PINNs by focusing on learning the underlying operators governing physical systems directly. Unlike traditional PINNs, which rely on solving specific PDEs or system dynamics using neural networks, operator learning (Lu et al. 2021a; Wang et al. 2021b) aims to learn mappings between input and output functions (such as solutions to PDEs), thus enabling more generalizable models that can be applied to a wide range of physical problems without requiring explicit physics-based constraints. The transition from PINNs to deep operator learning presents several exciting avenues for future research. First, the development of more robust training algorithms capable of learning high-dimensional operator mappings efficiently will be critical. These models should be designed to handle the complexities of multi-scale, multi-physics problems, which are often encountered in real-world applications. Additionally, there is a need for further research into the integration of data-driven and physics-based models in operator learning, particularly in cases where the available data is sparse or noisy. Future studies should also explore the interpretability and explainability of operator learning models to ensure their deployment in safety-critical engineering fields.

In summary, while PINNs have laid the groundwork for physics-informed learning, the transition toward deep operator learning offers exciting new possibilities for generalizing the framework to broader classes of problems. Continued research in this direction will be essential to unlocking the full potential of neural networks in scientific computing and engineering applications.

6 Conclusion

In conclusion, PINNs have emerged as a powerful tool for solving PDEs by embedding physical laws directly into the loss function of a neural network, thereby bridging the gap between traditional numerical methods and modern machine learning techniques.

This review paper provides a comprehensive examination of the advancements and applications of PINNs across a broad spectrum of scientific and engineering domains. Diverse innovative PINNs from different perspectives have been highlighted, including their architecture, adaptive resampling, loss and activation functions, feature embedding, and large models. Despite the successes of PINNs, the review also acknowledges ongoing challenges, such as the need for larger batch sizes, sophisticated training procedures, and the handling of high-dimensional and non-linear problems. Future research aims to address these issues, focusing on enhancing the scalability, interpretability, and reliability of PINNs.

PINNs represent a transformative technology at the intersection of machine learning and computational science. By embedding physical knowledge directly into machine learning models, they offer a powerful toolset for solving complex PDEs across multiple disciplines. As research progresses, it is anticipated that PINNs will continue to evolve, addressing existing limitations and expanding their applicability to new domains, thus contributing to the advancement of scientific discovery and engineering practice.

Acknowledgements The work in this paper is partially supported by Guangdong Innovation and Entrepreneurship Team Project under Grant 2021ZT09X070, Guangdong Zhujiang Leadership Project under Grant 2021CX02X011, and Chinese Huoju Talents Project under Grant 2023HJJH01.

Author contributions Kuang Luo, Jingshang Zhao, Yingping Wang, Jiayao Li, Junjie Wen, Jiong Liang and Shaolin Liao wrote the main manuscript text. All authors reviewed the manuscript.

Data availability The data in this study is available from the corresponding author upon reasonable request.

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Aarts LP, Veer P (2001) Neural network method for solving partial differential equations. Neural Process Lett 14(3):261–271. https://doi.org/10.1023/a:1012784129883

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mane D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viegas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2016) TensorFlow: large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467

Abbasi J, Andersen (2024) Physical activation functions (PAFs): an approach for more efficient induction of physics into physics-informed neural networks (PINNs). Neurocomputing 608:128352. https://doi.org/10.1016/j.neucom.2024.128352

Berblinger M, Schlier C (1991) Monte Carlo integration with quasi-random numbers: some experience. Comput Phys Commun 66(2–3):157–166. https://doi.org/10.1016/0010-4655(91)90064-r

Blechschmidt J, Ernst OG (2021) Three ways to solve partial differential equations with neural networks—a review. GAMM-Mitteilungen 44(2):e202100006. https://doi.org/10.1002/gamm.202100006

Bonev B, Kurth T, Hundt C, Pathak J, Baust M, Kashinath K, Anandkumar A (2023) Spherical Fourier neural operators: learning stable dynamics on the sphere. In: Proceedings of the 40th international conference on machine learning. ICML'23

Boussange V, Becker S, Jentzen A, Kuckuck B, Pellissier L (2023) Deep learning approximations for non-local nonlinear PDEs with Neumann boundary conditions. Partial Differ Equ Appl 4(6):51

Cai S, Mao Z, Wang Z, Yin M, Karniadakis GE (2021) Physics-informed neural networks (PINNs) for fluid mechanics: a review. Acta Mech Sin 37(12):1727–1738

Chen T, 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):911–917. https://doi.org/10.1109/72.392253

Chen Y, Lu L, Karniadakis GE, Dal Negro L (2020a) Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Opt Express 28(8):11618. https://doi.org/10.1364/oe.384875

Chen F, Sondak D, Protopapas P, Mattheakis M, Liu S, Agarwal D, Di Giovanni M (2020b) NeuroDiffEq: a Python package for solving differential equations with neural networks. J Open Source Softw 5(46):1931. https://doi.org/10.21105/joss.01931

Chernyshenko AY, Olshanskii MA (2015) An adaptive octree finite element method for PDEs posed on surfaces. Comput Methods Appl Mech Eng 291:146–172. https://doi.org/10.1016/j.cma.2015.03.025

Cho J, Nam S, Yang H, Yun S-B, Hong Y, Park E (2022) Separable PINN: mitigating the curse of dimensionality in physics-informed neural networks. arXiv:2211.08761 [cs.LG]

Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555 [cs.NE]

Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B, Bharath AA (2018) Generative adversarial networks: an overview. IEEE Signal Process Mag 35(1):53–65. https://doi.org/10.1109/msp.2017.2765202

Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 2(4):303–314. https://doi.org/10.1007/bf02551274

Daw A, Bu J, Wang S, Perdikaris P, Karpatne A (2023) Mitigating propagation failures in physics-informed neural networks using retain-resample-release (R3) sampling. In: Proceedings of the 40th international conference on machine learning. Proceedings of machine learning research, vol 202. pp 7264–7302. https://proceedings.mlr.press/v202/daw23a.html

Di Leoni PC, Lu L, Meneveau C, Karniadakis GE, Zaki TA (2023) Neural operator prediction of linear instability waves in high-speed boundary layers. J Comput Phys 474:111793

Diao Y, Yang J, Zhang Y, Zhang D, Du Y (2023) Solving multi-material problems in solid mechanics using physics-informed neural networks based on domain decomposition technology. Comput Methods Appl Mech Eng 413:116120. https://doi.org/10.1016/j.cma.2023.116120

Dick E (2009) Introduction to finite volume methods in computational fluid dynamics. In: Computational fluid dynamics. pp 275–301

Dissanayake M, Phan-Thien N (1994) Neural-network-based approximations for solving partial differential equations. Commun Numer Methods Eng 10(3):195–201. https://doi.org/10.1002/cnm.1640100303

El-Ajou A (2021) Adapting the Laplace transform to create solitary solutions for the nonlinear time-fractional dispersive PDEs via a new approach. Eur Phys J Plus 136(2):229

Fang Z (2022) A high-efficient hybrid physics-informed neural networks based on convolutional neural network. IEEE Trans Neural Netw Learn Syst 33(10):5514–5526. https://doi.org/10.1109/tnnls.2021.3070878

Fang Z, Zhan J (2020) Deep physical informed neural networks for metamaterial design. IEEE Access 8:24506–24513. https://doi.org/10.1109/access.2019.2963375

Faria RR, Capron B, Secchi AR, Souza MB Jr (2024) A data-driven tracking control framework using physics-informed neural networks and deep reinforcement learning for dynamical systems. Eng Appl Artif Intell 127:107256

Fu Z, Xu W, Liu S (2024) Physics-informed kernel function neural networks for solving partial differential equations. Neural Netw 172:106098. https://doi.org/10.1016/j.neunet.2024.106098

Gao Y, Ng MK (2022) Wasserstein generative adversarial uncertainty quantification in physics-informed neural networks. J Comput Phys 463:111270. https://doi.org/10.1016/j.jcp.2022.111270

Gao H, Sun L, Wang J-X (2021) PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. J Comput Phys 428:110079. https://doi.org/10.1016/j.jcp.2020.110079

Giles CL, Kuhn GM, Williams RJ (1994) Dynamic recurrent neural networks: theory and applications. IEEE Trans Neural Netw 5(2):153–156. https://doi.org/10.1109/tnn.1994.8753425

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, vol 27

Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, Chen T (2018) Recent advances in convolutional neural networks. Pattern Recogn 77:354–377. https://doi.org/10.1016/j.patcog.2017.10.013

Gu Y, Yang H, Zhou C (2021) SelectNet: self-paced learning for high-dimensional partial differential equations. J Comput Phys 441:110444. https://doi.org/10.1016/j.jcp.2021.110444

Guan W, Yang K, Chen Y, Liao S, Guan Z (2023) A dimension-augmented physics-informed neural network (DaPINN) with high level accuracy and efficiency. J Comput Phys 491:112360. https://doi.org/10.1016/j.jcp.2023.112360

Guo H, Zhuang X, Chen P, Alajlan N, Rabczuk T (2022) Analysis of three-dimensional potential problems in non-homogeneous media with physics-informed deep collocation method using material transfer learning and sensitivity analysis. Eng Comput 38(6):5423–5444. https://doi.org/10.1007/s00366-022-01633-6

Hafiz AM, Faiq I, Hassaballah M (2024) Solving partial differential equations using large-data models: a literature review. Artif Intell Rev 57(6):152. https://doi.org/10.1007/s10462-024-10784-5

Haghighat E, Juanes R (2021a) SciANN: a Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. Comput Methods Appl Mech Eng 373:113552. https://doi.org/10.1016/j.cma.2020.113552

Haghighat E, Raissi M, Moure A, Gomez H, Juanes R (2021b) A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Comput Methods Appl Mech Eng 379:113741. https://doi.org/10.1016/j.cma.2021.113741

Halton JH (1960) On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer Math 2(1):84–90. https://doi.org/10.1007/bf01386213

Hanna JM, Aguado JV, Comas-Cardona S, Askri R, Borzacchiello D (2022) Residual-based adaptivity for two-phase flow simulation in porous media using physics-informed neural networks. Comput Methods Appl Mech Eng 396:115100

He Q, Barajas-Solano D, Tartakovsky G, Tartakovsky AM (2020) Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. Adv Water Resour 141:103610. https://doi.org/10.1016/j.advwatres.2020.103610

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Hu Z, Jagtap AD, Karniadakis GE, Kawaguchi K (2022) When do extended physics-informed neural networks (XPINNs) improve generalization? SIAM J Sci Comput 44(5):3158–3182. https://doi.org/10.1137/21m1447039

Hwang R, Lee JY, Shin JY, Hwang HJ (2022) Solving PDE-constrained control problems using operator learning. Proc AAAI Conf Artif Intell 36:4504–4512

Innerberger M, Praetorius D (2023) MooAFEM: an object oriented MATLAB code for higher-order adaptive fem for (nonlinear) elliptic PDEs. Appl Math Comput 442:127731

Jacot A, Gabriel F, Hongler C (2018) Neural tangent kernel: convergence and generalization in neural networks. In: Advances in neural information processing systems, vol 31

Jagtap AD, Kharazmi E, Karniadakis GE (2020a) Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. Comput Methods Appl Mech Eng 365:113028. https://doi.org/10.1016/j.cma.2020.113028

Jagtap AD, Kawaguchi K, Karniadakis GE (2020b) Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. J Comput Phys 404:109136. https://doi.org/10.1016/j.jcp.2019.109136

Jeong Y, Jo J, Lee T, Yoo J (2024) Combined analysis of thermofluids and electromagnetism using physics-informed neural networks. Eng Appl Artif Intell 133:108216. https://doi.org/10.1016/j.engappai.2024.108216

Jiao Y, Li D, Lu X, Yang JZ, Yuan C (2024) A Gaussian mixture distribution-based adaptive sampling method for physics-informed neural networks. Eng Appl Artif Intell 135:108770. https://doi.org/10.1016/j.engappai.2024.108770

Jo J, Jeong Y, Kim J, Yoo J (2024) Thermal conductivity estimation using physics-informed neural networks with limited data. Eng Appl Artif Intell 137:109079. https://doi.org/10.1016/j.engappai.2024.109079

Kan S, Cen Y, He Z, Zhang Z, Zhang L, Wang Y (2019) Supervised deep feature embedding with handcrafted feature. IEEE Trans Image Process 28(12):5809–5823. https://doi.org/10.1109/tip.2019.2901407

Karniadakis A, Em G (2020) Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Commun Comput Phys 28(5):2002–2041. https://doi.org/10.4208/cicp.oa-2020-0164

Kissas G, Yang Y, Hwuang E, Witschey WR, Detre JA, 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 358:112623. https://doi.org/10.1016/j.cma.2019.112623

Kollmannsberger S et al (2019) The finite cell method: towards engineering applications. https://books.google.com.tw/books?id=oJtdzQEACAAJ

Kolman R, Okrouhlík M, Berezovski A, Gabriel D, Kopačka J, Plešek J (2017) B-spline based finite element method in one-dimensional discontinuous elastic wave propagation. Appl Math Model 46:382–395

Krishnapriyan AS, Gholami A, Zhe S, Kirby RM, Mahoney MW (2021) Characterizing possible failure modes in physics-informed neural networks. In: Proceedings of the 35th international conference on neural information processing systems. NIPS '21. Curran Associates Inc., Red Hook, NY, USA

Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, vol 25

Kumar M, Yadav N (2011) Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey. Comput Math Appl 62(10):3796–3811. https://doi.org/10.1016/j.camwa.2011.09.028

Lagaris IE, Likas A, Fotiadis DI (1998) Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw 9(5):987–1000. https://doi.org/10.1109/72.712178

Lau GKR, Hemachandra A, Ng S-K, Low BKH (2024) PINNACLE: PINN adaptive collocation and experimental points selection. In: The Twelfth international conference on learning representations

Le-Duc T, Lee S, Nguyen-Xuan H, Lee J (2024) A hierarchically normalized physics-informed neural network for solving differential equations: application for solid mechanics problems. Eng Appl Artif Intell 133:108400. https://doi.org/10.1016/j.engappai.2024.108400

Lei G, Lei Z, Shi L, Zeng C, Zhou D-X (2023) Solving PDEs on spheres with physics-informed convolutional neural networks. arXiv:2308.09605 [math.NA]

Li Z, Liu F, Yang W, Peng S, Zhou J (2022) A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans Neural Netw Learn Syst 33(12):6999–7019. https://doi.org/10.1109/tnnls.2021.3084827

Li X, Deng J, Wu J, Zhang S, Li W, Wang Y-G (2024a) Physical informed neural networks with soft and hard boundary constraints for solving advection-diffusion equations using Fourier expansions. Comput Math Appl 159:60–75. https://doi.org/10.1016/j.camwa.2024.01.021

Li W, Liu Z, Chen K, Chen H, Liang S, Zou Z, Shi Z (2024b) DeepPhysiNet: bridging deep learning and atmospheric physics for accurate and continuous weather modeling. arXiv:2401.04125 [physics.ao-ph]

Li X, Wu J, Tai X, Xu J, Wang Y-G (2024c) Solving a class of multi-scale elliptic PDEs by Fourier-based mixed physics informed neural networks. J Comput Phys 508:113012. https://doi.org/10.1016/j.jcp.2024.113012

Lipton ZC, Berkowitz J, Elkan C (2015) A critical review of recurrent neural networks for sequence learning. arXiv:1506.00019 [cs.LG]

Liu WK, Li S, Park HS (2022) Eighty years of the finite element method: birth, evolution, and future. Arch Comput Methods Eng 29(6):4431–4453

Liu J-P, Wang B-Z, Chen C-S, Wang R (2024a) Inverse design method for horn antennas based on knowledge-embedded physics-informed neural networks. IEEE Antennas Wirel Propag Lett 23(6):1665–1669. https://doi.org/10.1109/lawp.2024.3365690

Liu Z, Wang Y, Vaidya S, Ruehle F, Halverson J, Soljačić M, Hou TY, Tegmark M (2024b) KAN: Kolmogorov-Arnold networks. https://arxiv.org/abs/2404.19756

Liu S, Protopapas P, Sondak D, Chen F (2025) Recent advances of NeuroDiffEq—an open-source library for physics-informed neural networks. arXiv Preprint. arXiv:2502.12177

Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE (2021a) Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat Mach Intell 3(3):218–229. https://doi.org/10.1038/s42256-021-00302-5

Lu L, Meng X, Mao Z, Karniadakis GE (2021b) DeepXDE: a deep learning library for solving differential equations. SIAM Rev 63(1):208–228. https://doi.org/10.1137/19m1274067

Luo K, Liao S, Guan Z, Liu B (2025) An enhanced hybrid adaptive physics-informed neural network for forward and inverse PDE problems. Appl Intell 55(4):255

Mallikarjunaiah S (2023) A deep learning feed-forward neural network framework for the solutions to singularly perturbed delay differential equations. Appl Soft Comput 148:110863

Mao Z, Jagtap AD, Karniadakis GE (2020) Physics-informed neural networks for high-speed flows. Comput Methods Appl Mech Eng 360:112789. https://doi.org/10.1016/j.cma.2019.112789

Mavi A, Bekar AC, Haghighat E, Madenci E (2023) An unsupervised latent/output physics-informed convolutional-LSTM network for solving partial differential equations using peridynamic differential operator. Comput Methods Appl Mech Eng 407:115944. https://doi.org/10.1016/j.cma.2023.115944

McClenny LD, Braga-Neto UM (2023) Self-adaptive physics-informed neural networks. J Comput Phys 474:111722. https://doi.org/10.1016/j.jcp.2022.111722

McClenny LD, Haile MA, Braga-Neto UM (2021) TensorDiffEq: Scalable multi-GPU forward and inverse solvers for physics informed neural networks. arXiv Preprint. arXiv:2103.16034

Mckay MD, Beckman RJ, Conover WJ (2000) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 42(1):55–61. https://doi.org/10.1080/00401706.2000.10485979

Melchers H, Crommelin D, Koren B, Menkovski V, Sanderse B (2023) Comparison of neural closure models for discretised PDEs. Comput Math Appl 143:94–107

Mildenhall B, Srinivasan P, Tancik M, Barron J, Ramamoorthi R, Ng R (2020) NeRF: representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. pp 405–421

Monterola C, Saloma C (2003) Solving the nonlinear Schrödinger equation with an unsupervised neural network: estimation of error in solution. Opt Commun 222(1–6):331–339. https://doi.org/10.1016/s0030-4018(03)01570-0

Mowlavi S, Nabi S (2023) Optimal control of PDEs using physics-informed neural networks. J Comput Phys 473:111731

Mugler D, Scott R (1988) Fast Fourier transform method for partial differential equations, case study: the 2-D diffusion equation. Comput Math Appl 16(3):221–228

Nabian MA, Gladstone RJ, Meidani H (2021) Efficient training of physics-informed neural networks via importance sampling. Comput-Aided Civ Infrastruct Eng 36(8):962–977. https://doi.org/10.1111/mice.12685

Namaki N, Eslahchi M, Salehi R (2023) The use of physics-informed neural network approach to image restoration via nonlinear PDE tools. Comput Math Appl 152:355–363

Norambuena A, Mattheakis M, González FJ, Coto R (2024) Physics-informed neural networks for quantum control. Phys Rev Lett 132(1):010801. https://doi.org/10.1103/physrevlett.132.010801

Oñate E, Owen R (2011) Particle-based methods: fundamentals and applications, vol 25. https://doi.org/10.1007/978-94-007-0735-1

Pan Y-Q, Wang R, Wang B-Z (2024) Physics-informed neural networks with embedded analytical models: inverse design of multilayer dielectric-loaded rectangular waveguide devices. IEEE Trans Microw Theory Tech 72(7):3993–4005. https://doi.org/10.1109/tmtt.2023.3343028

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: an imperative style, high-performance deep learning library. In: Advances in neural information processing systems, vol 32

Patel R, Bhartiya S, Gudi R (2023) Optimal temperature trajectory for tubular reactor using physics informed neural networks. J Process Control 128:103003

Patil R, Boit S, Gudivada V, Nandigam J (2023) A survey of text representation and embedding techniques in NLP. IEEE Access 11:36120–36146. https://doi.org/10.1109/access.2023.3266377

Peng W, Zhou W, Zhang J, Yao W (2020) Accelerating physics-informed neural network training with prior dictionaries. https://arxiv.org/abs/2004.08151

Peng W, Zhang J, Zhou W, Zhao X, Yao W, Chen X (2021) IDRLNet: a physics-informed neural network library. https://doi.org/10.48550/ARXIV.2107.04320

Perez L, Wang J (2017) The effectiveness of data augmentation in image classification using deep learning. https://arxiv.org/abs/1712.04621

Qin D, Du Y, Liu B, Huang W (2019b) A b-spline finite element method for nonlinear differential equations describing crystal surface growth with variable coefficient. Adv Differ Equ 2019:1–16

Rahaman N, Baratin A, Arpit D, Draxler F, Lin M, Hamprecht F, Bengio Y, Courville A (2019) On the spectral bias of neural networks. In: Proceedings of the 36th international conference on machine learning. Proceedings of machine learning research, vol 97. pp 5301–5310. https://proceedings.mlr.press/v97/rahaman19a.html

Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707. https://doi.org/10.1016/j.jcp.2018.10.045

Ranade R, Hill C, Pathak J (2021) DiscretizationNet: a machine-learning based solver for Navier-Stokes equations using finite volume discretization. Comput Methods Appl Mech Eng 378:113722

Rao C, Sun H, Liu Y (2021) Physics-informed deep learning for computational elastodynamics without labeled data. J Eng Mech. https://doi.org/10.1061/(asce)em.1943-7889.0001947

Ren P, Rao C, Liu Y, Wang J-X, Sun H (2022) PhyCRNet: physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. Comput Methods Appl Mech Eng 389:114399. https://doi.org/10.1016/j.cma.2021.114399

Rigas S, Papachristou M, Papadopoulos T, Anagnostopoulos F, Alexandridis G (2024) Adaptive training of grid-dependent physics-informed Kolmogorov-Arnold networks. arXiv:2407.17611 [cs.LG]

Rodkina A, Kelly C (2011) Stochastic difference equations and applications. Springer, pp 1517–1520

Santos FD, Akhound-Sadegh T, Ravanbakhsh S (2023) Physics-informed transformer networks. In: the symbiosis of deep learning and differential equations III. https://openreview.net/forum?id=zu80h9YryU

Schiassi E, D'Ambrosio A, Furfaro R (2022) Bellman neural networks for the class of optimal control problems with integral quadratic cost. IEEE Trans Artif Intell 5(3):1016–1025

Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681. https://doi.org/10.1109/78.650093

Sedykh A, Podapaka M, Sagingalieva A, Pinto K, Pflitsch M, Melnikov A (2024) Hybrid quantum physics-informed neural networks for simulating computational fluid dynamics in complex shapes. Mach Learn Sci Technol 5(2):025045. https://doi.org/10.1088/2632-2153/ad43b2

Shukla K, Toscano JD, Wang Z, Zou Z, Karniadakis GE (2024) A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks. Comput Methods Appl Mech Eng 431:117290. https://doi.org/10.1016/j.cma.2024.117290

Sobol' IM (1967) On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput Math Math Phys 7(4):86–112. https://doi.org/10.1016/0041-5553(67)90144-9

Soldatenko S, Yusupov R (2017) Predictability in deterministic dynamical systems with application to weather forecasting and climate modelling. In: Dynamical systems-analytical and computational techniques, vol 101

Song Y, Wang H, Yang H, Taccari ML, Chen X (2024) Loss-attentional physics-informed neural networks. J Comput Phys 501:112781

Stein M (1987) Large sample properties of simulations using Latin hypercube sampling. Technometrics 29(2):143–151. https://doi.org/10.1080/00401706.1987.10488205

Sun L, Gao H, Pan S, Wang J-X (2020) Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. Comput Methods Appl Mech Eng 361:112732

Tancik M, Srinivasan P, Mildenhall B, Fridovich-Keil S, Raghavan N, Singhal U, Ramamoorthi R, Barron J, Ng R (2020) Fourier features let networks learn high frequency functions in low dimensional domains. Adv Neural Inf Process Syst 33:7537–7547

Tang K, Wan X, Yang C (2023) DAS-PINNs: a deep adaptive sampling method for solving high-dimensional partial differential equations. J Comput Phys 476:111868. https://doi.org/10.1016/j.jcp.2022.111868

Tang K, Zhai J, Wan X, Yang C (2024) Adversarial adaptive sampling: unify PINN and optimal transport for the approximation of PDEs. In: The twelfth international conference on learning representations. https://openreview.net/forum?id=7QI7tVrh2c

Taylor JM, Pardo D, Muga I (2023) A deep Fourier residual method for solving PDEs using neural networks. Comput Methods Appl Mech Eng 405:115850

Tripura T, Chakraborty S (2023) Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. Comput Methods Appl Mech Eng 404:115783. https://doi.org/10.1016/j.cma.2022.115783

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser LU, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, vol 30

Wandel N, Weinmann M, Neidlin M, Klein R (2022) Spline-PINN: approaching PDEs without data using fast, physics-informed Hermite-spline CNNs. Proc AAAI Conf Artif Intell 36(8):8529–8538. https://doi.org/10.1609/aaai.v36i8.20830

Wang Y, Zhong L (2024) NAS-PINN: neural architecture search-guided physics-informed neural network for solving PDEs. J Comput Phys 496:112603. https://doi.org/10.1016/j.jcp.2023.112603

Wang S, Wang H, Perdikaris P (2021a) On the eigenvector bias of Fourier feature networks: from regression to solving multi-scale PDEs with physics-informed neural networks. Comput Methods Appl Mech Eng 384:113938. https://doi.org/10.1016/j.cma.2021.113938

Wang S, Wang H, Perdikaris P (2021b) Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Sci Adv. https://doi.org/10.1126/sciadv.abi8605

Wang S, Teng Y, Perdikaris P (2021c) Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM J Sci Comput 43(5):3055–3081. https://doi.org/10.1137/20m1318043

Wang S, Yu X, Perdikaris P (2022a) When and why PINNs fail to train: a neural tangent kernel perspective. J Comput Phys 449:110768. https://doi.org/10.1016/j.jcp.2021.110768

Wang C, Li S, He D, Wang L (2022b) Is [CDATA[L^2]]L^2 physics informed loss always suitable for training physics informed neural network? Adv Neural Inf Process Syst 35:8278–8290

Wang Y, Han X, Chang C-Y, Zha D, Braga-Neto U, Hu X (2022c) Auto-PINN: understanding and optimizing physics-informed neural architecture. arXiv:2205.13748 [cs.LG]

Wang J, Mo YL, Izzuddin B, Kim C-W (2023) Exact Dirichlet boundary physics-informed neural network EPINN for solid mechanics. Comput Methods Appl Mech Eng 414:116184. https://doi.org/10.1016/j.cma.2023.116184

Wang Y, Yao Y, Guo J, Gao Z (2024a) A practical PINN framework for multi-scale problems with multi-magnitude loss terms. J Comput Phys 510:113112. https://doi.org/10.1016/j.jcp.2024.113112

Wang S, Li B, Chen Y, Perdikaris P (2024b) PirateNets: physics-informed deep learning with residual adaptive networks. arXiv:2402.00326 [cs.LG]

Wang Y, Sun J, Bai J, Anitescu C, Eshaghi MS, Zhuang X, Rabczuk T, Liu Y (2024c) Kolmogorov Arnold informed neural network: a physics-informed deep learning framework for solving forward and inverse problems based on Kolmogorov Arnold networks. arXiv:2406.11045 [cs.LG]

Wang Z, Zhang L, Zhang Z, Xu Z-QJ (2024d) Loss jump during loss switch in solving PDEs with neural networks. arXiv:2405.03095 [cs.LG]

Wong T-T, Luk W-S, Heng P-A (1997) Sampling with Hammersley and Halton points. J Graph Tools 2(2):9–24. https://doi.org/10.1080/10867651.1997.10487471

Wong JC, Ooi CC, Gupta A, Ong Y-S (2024) Learning in sinusoidal spaces with physics-informed neural networks. IEEE Trans Artif Intell 5(3):985–1000. https://doi.org/10.1109/tai.2022.3192362

Wu W, Feng X, Xu H (2022) Improved deep neural networks with domain decomposition in solving partial differential equations. J Sci Comput. https://doi.org/10.1007/s10915-022-01980-y

Wu C, Zhu M, Tan Q, Kartha Y, Lu L (2023) A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. Comput Methods Appl Mech Eng 403:115671. https://doi.org/10.1016/j.cma.2022.115671

Wu H, Luo H, Wang H, Wang J, Long M (2024) Transolver: a fast transformer solver for PDEs on general geometries. In: Forty-first international conference on machine learning. https://openreview.net/forum?id=Ywl6pODXjB

Xiang Z, Peng W, Liu X, Yao W (2022) Self-adaptive loss balanced physics-informed neural networks. Neurocomputing 496:11–34. https://doi.org/10.1016/j.neucom.2022.05.015

Xiao Y, Yang LM, Shu C, Chew SC, Khoo BC, Cui YD, Liu YY (2024) Physics-informed quantum neural network for solving forward and inverse problems of partial differential equations. Phys Fluids. https://doi.org/10.1063/5.0226232

Yang L, Zhang D, Karniadakis GE (2020) Physics-informed generative adversarial networks for stochastic differential equations. SIAM J Sci Comput 42(1):292–317. https://doi.org/10.1137/18m1225409

Yang Z, Qiu Z, Fu D (2023) DMIS: dynamic mesh-based importance sampling for training physics-informed neural networks. Proc AAAI Conf Artif Intell 37(4):5375–5383. https://doi.org/10.1609/aaai.v37i4.25669

Yu Y, Si X, Hu C, Zhang J (2019) A review of recurrent neural networks: LSTM cells and network architectures. Neural Comput 31(7):1235–1270. https://doi.org/10.1162/neco_a_01199

Yu J, Lu L, Meng X, Karniadakis GE (2022) Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. Comput Methods Appl Mech Eng 393:114823

Yuan B, Wang H, Heitor A, Chen X (2024) f-PICNN: a physics-informed convolutional neural network for partial differential equations with space-time domain. J Comput Phys 515:113284. https://doi.org/10.1016/j.jcp.2024.113284

Zeng S, Zhang Z, Zou Q (2022) Adaptive deep neural networks methods for high-dimensional partial differential equations. J Comput Phys 463:111232. https://doi.org/10.1016/j.jcp.2022.111232

Zhang F, Yuan NJ, Lian D, Xie X, Ma W-Y (2016) Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp 353–362. https://doi.org/10.1145/2939672.2939673

Zhang E, Dao M, Karniadakis GE, Suresh S (2022a) Analyses of internal structures and defects in materials using physics-informed neural networks. Sci Adv 8(7):1. https://doi.org/10.1126/sciadv.abk0644

Zhang Z, Cutkosky A, Paschalidis I (2022b) PDE-based optimal strategy for unconstrained online learning. In: International conference on machine learning. PMLR, pp 26085–26115

Zhang Z, Yan X, Liu P, Zhang K, Han R, Wang S (2023) A physics-informed convolutional neural network for the simulation and prediction of two-phase darcy flows in heterogeneous porous media. J Comput Phys 477:111919

Zhao Z, Ding X, Prakash BA (2024) PINNsformer: a transformer-based framework for physics-informed neural networks. In: The twelfth international conference on learning representations. https://openreview.net/forum?id=DO2WFXU1Be

Zheng Y, Hu C, Wang X, Wu Z (2023) Physics-informed recurrent neural network modeling for predictive control of nonlinear processes. J Process Control 128:103005