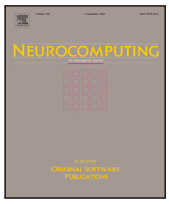
Neurocomputing 648 (2025) 130518

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Survey Paper

# Architectures, variants, and performance of neural operators: A comparative review

Shengjun Liu [a], Yu Yu [a], Ting Zhang [a], Hanchao Liu [a,*], Xinru Liu [a], Deyu Meng [b]

[a] School of Mathematics and Statistics, Central South University, Changsha, 410083, China
[b] Research Institute for Mathematics and Mathematical Technology, Xi'an Jiaotong University, Xi'an, 710049, China

ARTICLE INFO

Communicated by R. Prevete

Dataset link: https://github.com/YuYu8900/Neural-Operators

*Keywords:*
AI for science
Partial differential equations
Neural operators
Complex systems
Physical information
Operator basis

ABSTRACT

In recent years, neural operators have emerged as effective alternatives to traditional numerical solvers. They are known for their efficient computation, excellent generalization, and high solving accuracy. Many researchers have shown interest in their design and application. This paper provides a comprehensive summary and analysis of neural operators. We categorize them into three types based on their architecture: deep operator networks (DeepONets), integral kernel operators, and transformer-based neural operators. We then discuss the basic structures and properties of these operator types. Furthermore, we summarize and discuss the various variants and extensions of these three types of neural operators from three directions: (1) operator basis-based neural operator variants; (2) physics-informed neural operator variants; and (3) application of neural operator variants in complex systems. We also analyze the characteristics and performance of different operator methods through numerical experiments. Taking into account these discussions and analyses, we provide perspectives and suggestions regarding the challenges and potential enhancements for different neural operators. This offers valuable guidance and suggestions for the practical application and development of neural operators.

## 1. Introduction

Partial differential equations (PDEs) find wide application in simulating complex phenomena across physics, chemistry, biology, and engineering domains [1–3]. Initially, traditional numerical methods are commonly employed for solving PDEs, such as the finite difference method [4] and the finite element method [5]. However, these traditional methods heavily rely on grid discretization, particularly when dealing with high-dimensional PDEs that necessitate numerous grid nodes and expensive calculations. Moreover, traditional numerical solvers often pose a significant inconvenience by requiring a complete restart whenever even slight changes are made to the equation parameters. This limitation significantly restricts the usability of traditional methods. The widespread application of neural networks in recent years has drawn researchers' attention because of their powerful universal approximation capabilities [6,7]. Neural networks emerge as potential alternatives to traditional numerical solvers [8–12]. In contrast to traditional numerical methods, neural networks (NNs) possess learning capabilities and adaptability, enabling them to approximate highly complex nonlinear functions.

Previously, a mainstream approach is physics-informed neural networks (PINNs) [13–15], which incorporate known physical laws into the network structure and loss function to integrate seamlessly data and mathematical physics models, even in partially understood, uncertain and high-dimensional contexts. PINNs have demonstrated remarkable success in solving high-dimensional problems, partly due to their ability to overcome the curse of dimensionality. Similar to classical methods, PINNs are limited to solving a single PDE. In the event of any changes to the parameters of the partial differential equation, these methods require retraining a new neural network. Moreover, in many natural processes, the underlying physics may remain unknown or highly complex, posing challenges to physics-informed modeling and control. Further, the speed and accuracy of the aforementioned methods are significantly lower when compared to classical approaches, particularly in tackling low-dimensional problems.

Building upon the aforementioned limitations of PINNs, Lu et al. [16] introduced the Deep Operator Network (DeepONet), which learns mappings between infinite-dimensional function spaces from datasets. Neural operators (NOs) learn to solve an entire family of PDEs, in contrast to classical methods, which solve one instance of the equation. For any example within a family of parametric PDEs, training only needs to be performed once to obtain the corresponding solution. Moreover, neural operators are capable of addressing problems with

* Corresponding author.
*E-mail addresses:* shjliu.cg@csu.edu.cn (S. Liu), yuyu2022@csu.edu.cn (Y. Yu), 212101002@csu.edu.cn (T. Zhang), hchliu@csu.edu.cn (H. Liu), liuxinru@csu.edu.cn (X. Liu), dymeng@mail.xjtu.edu.cn (D. Meng).

https://doi.org/10.1016/j.neucom.2025.130518
Received 8 January 2025; Received in revised form 3 May 2025; Accepted 12 May 2025
Available online 6 June 2025

diverse boundary conditions and intricate geometries [17,18]. Several distinct architectures, such as DeepONet [16], integral kernel operator [19,20], and transformer-based operator architectures [21], have been proposed for operator learning. These architectures have demonstrated the capability to efficiently and precisely compute solutions to complex systems governed by PDEs and have found extensive applications. Nevertheless, neural operators do have certain limitations. For instance, neural operators typically require large amounts of training data, particularly for complex PDEs, which significantly restricts their applicability. Despite the numerous limitations and challenges associated with neural operators, ongoing research continues to propose new and enhanced approaches owing to their significant potential in solving PDEs.

On the other hand, neural operators are essentially mapping between learning function spaces, so they have wide application prospects in various fields. For example, in the field of modeling the spread of recent fires, Zheng et al. [22] proposed diffusion model sampling with neural operator (DSNO), an efficient method to solve the probability flow differential equations, to accelerate the sampling process of diffusion models. Liao et al. [23] introduced the Score Neural Operator, a novel generative modeling framework that learns to map multiple probability distributions to their corresponding score functions. This approach enables generalization to unseen distributions without retraining, marking a significant advancement in the field. Similarly, in the computer vision direction, Wei et al. [24] proposed Super-resolution Neural Operator (SRNO), a deep operator learning framework that treats LR(low-resolution)-HR(high-resolution) image pairs as continuous functions approximated with different grid sizes and learns the mapping between the corresponding function spaces to resolve high-resolution images at arbitrary scales from their low-resolution counterparts. Han et al. [25] proposed an inference-time adaptive network width optimization method for arbitrary scale super-resolution modules, dubbed as Scalable Super-Resolution Neural Operator (SSRNO), which is capable of efficient performance preserving deployment on various mobile or edge devices with only a user input parameter indicating the desired compression rate. Neural ordinary differential equations (NODEs), a pivotal work in differential equation-inspired deep learning, generalize residual networks and are applied to tasks such as image and time series classification, as well as image generation. Cho et al. [26] presented a neural operator-based method called branched Fourier neural operator (BFNO), to define the time-derivative term. Neural operators also have great potential in other areas of machine learning and artificial intelligence. Pal et al. [27] offered an operator theoretic reformulation of the INR model, which is called Operator INR (or O-INR).

Prior to this, previous works only provided summaries and discussions on specific integral kernel operator frameworks or only compared DeepONet and FNO [28,29], leaving a gap in terms of a comprehensive overview of the overall neural operator frameworks. In this paper, we provide a thorough analysis of the progress made in neural operators from different bottom-level development directions. It aims to integrate and systematize the existing research findings while also uncovering the interconnections and disparities among various neural operator frameworks. We primarily focus on discussing the network architecture of neural operators for solving PDEs and their various extensions and variants that have found wide applications in real-world problems. Fig. 1 illustrates the schematic representation of the historical development of neural operators. In Section 2, we introduce representative neural operator network architectures, including DeepONets, integral kernel operators (Graph Neural Operator (GNO), Fourier Neural Operator (FNO)), and transformer-based neural operators (Galerkin-Transformer (GK-Transformer), General Neural Operator Transformer (GNOT)). We thoroughly discuss their respective advantages and limitations. In Section 3, we explore different variants of neural operators in three general directions: operator basis-based, physics-informed, and complex systems related. We examine the properties and performance of various operator variants through experimental analysis. Lastly, in Section 4, we summarize the evolution and limitations of neural operators and offer insights into future research directions, which strives to offer systematic and practical guidance for future research in the field of neural operators.

## 2. PDEs and neural operators

The general form of the nonlinear coupled PDE equation can be described as:

$$\begin{aligned} \mathcal{F}(x,a,u) &= f, \quad x \in \Omega, \\ \mathcal{B}(u) &= u_b, \quad x \in \partial\Omega, \\ \mathcal{I}(u) &= u_0, \quad x \in \Omega \times \{T_0\}, \end{aligned} \tag{1}$$

where $a$, $f$ represent parameter functions and control functions, and $u_0$ and $u_b$ are the corresponding initial and boundary conditions. The goal of neural operators is to learn mappings $(x, u_0, u_b, a, f) \mapsto u$ from the above systems.

Take 2D Darcy flow equation as an example,

$$\begin{aligned} -\nabla \cdot (a(x)\nabla u(x)) &= f(x), \quad x \in (0,1)^2, \\ u(x) &= 0, \quad x \in \partial(0,1)^2. \end{aligned} \tag{2}$$

The above formula corresponds to the general form Eq. (1). The goal is to learn the operator: $a \mapsto u$. For the convenience of the subsequent discussion, we mainly take the mapping between the parameter function $a$ and the solution function $u$: $a \mapsto u$ as an example.

Given the existing dataset $D = \{a_i, u_i\}_{1 \le i \le N}$, and $u_i = \tilde{G}(a_i)$, the objective of neural operators can be expressed as follows:

$$\min_{\theta \in R^p} \left\| \tilde{G}(a) - G_\theta(a) \right\|. \tag{3}$$

Here, $G$ represents the neural operator, and $\theta$ denotes learnable parameters, and $\tilde{G}$ represents the operator mapping between function spaces that need to be learned. The optimal parameters can be obtained by minimizing the loss function:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left\| u_i - G_\theta(a_i) \right\|^2, \tag{4}$$

where $u_i$ represents the ground truth solution, and $N$ denotes the total number of samples in the dataset.

During the process of solving PDEs, neural operators can be broadly categorized into three groups based on their architectures: DeepONets [16,30] based on the universal approximation theorem; mesh-based and graph-based integral kernel operators [19,20]; and transformer-based neural operators [21,31]. The new neural operators proposed by subsequent researchers are mainly based on the modification and extension of these three operator structures. This section will introduce the three fundamental neural operators for solving PDEs.

### 2.1. DeepONet

Lu et al. [16] first proposed DeepONet to learn operator mapping based on the universal approximation theorem [6]. The fundamental idea behind DeepONet is to learn the operator $G_\theta(a)$, where independent variables in PDEs (such as spatial coordinates and forcing terms $f(x)$) serve as inputs of DeepONet, and the solution of PDEs is the output. As shown in Fig. 2, the conventional DeepONet architecture comprises two main components: a trunk network and a branch network. The architecture can be expressed in the following form:

$$G_\theta(a)(x) = \sum_{k=1}^{p} b_k(a) t_k(x) + b_0, \tag{5}$$

where $G_\theta(a)$ is the operator that approximates the mapping between the input function $a$ and the solution of the equation, bias $b_0 \in R$ is the learnable parameter, $b_k$ and $t_k$ respectively represent outputs of the branch network and the trunk network, $y$ represents the query


**Fig. 1.** The history of neural operator development.

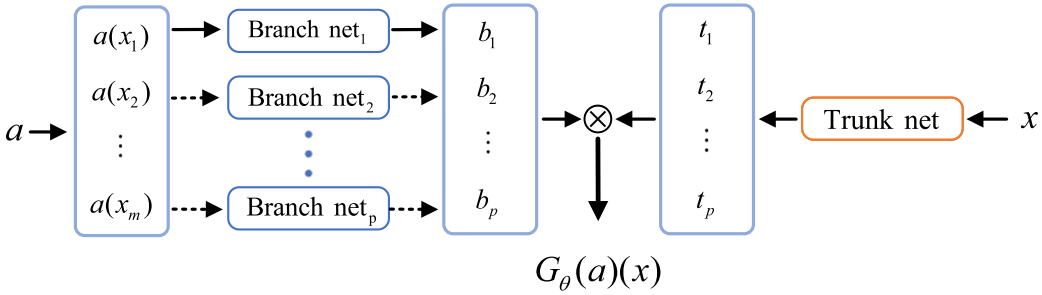



**Fig. 2.** DeepONet network architecture: DeepONet has a trunk network and p-stacked branch networks. It can be seen that the combination of branch and trunk networks is also free. We can use a single branch to share learning parameters with trunk networks or also can use multiple branch or trunk networks to match. This type is common in multi-input or multi-scale DeepONet [32,33].

coordinate points. The choice for branch networks and trunk networks is flexible and depends on the specific problem and the nature of the PDEs being solved. Common options for the basic neural network architectures include feedforward neural networks (FNNs) or convolutional neural networks (CNNs).

DeepONet is a simple yet powerful network architecture that leverages the universal approximation capabilities of neural networks to effectively and efficiently learn operators from datasets, thereby enabling accurate results. DeepONet is mesh-independent, enabling it to handle complex PDE problems without relying on explicitly defined meshes or discretizations. However, it is crucial to note that the performance and applicability of DeepONet, as a data-driven deep learning method, depend on factors such as the quality and quantity of the training data, the selected neural network structure, and the tuning of hyperparameters. Moreover, the input function of the branch network must be discretized within a finite-dimensional space using a limited number of points called sensors to make use of it. However, this discretization process can result in performance degradation when addressing high-dimensional or complex problems.

### 2.2. Integral kernel operator

Li et al. [20] were the pioneers in proposing the graph neural operator (GNO) for operator learning using graph neural networks [34] and Nyström's approximation theorem [35]. Afterward, they introduced a general architecture known as the integral kernel operator, and subsequent studies focused on exploring the design of the integral kernel within the linear operator layer. For example, they explored using the Fourier transform in the spectral domain to parameterize the integral operator layer and design a Fourier neural operator (FNO).

The integral kernel operator utilizes a composition of a linear integral operator $\mathcal{K}$ and a pointwise non-linear activation function $\sigma$ to effectively approximate highly non-linear operators [28]. The architecture is defined as:

$$G_\theta := \mathcal{Q} \circ \sigma_T \left(W_{T-1} + \mathcal{K}_{T-1} + b_{T-1}\right) \circ \cdots \circ \sigma_1 \left(W_0 + \mathcal{K}_0 + b_0\right) \circ \mathcal{P}, \tag{6}$$

where $\mathcal{P}$ and $\mathcal{Q}$ represent local lifting and projection mapping operations, respectively. $W_t$ represent local linear operators, $\mathcal{K}_t$ represent integral kernel operators, and $b_t$ represent bias functions. After performing a lifting operation on the integral kernel to get $v_0$, one iterates through $T$ layers of the integral kernel, progressing from $v_0$ to $v_1$ to … to $v_T$. Finally, a local transformation $\mathcal{Q} : \mathcal{R}^{d_{v_t}} \to \mathcal{R}^{d_u}$ is used to project $v_T$ back into the output domain.

The integral kernel operator is defined in the following Eq. (7):

$$\left(\mathcal{K} v_t\right)(x) = \int_D \kappa^{(t)}(x, y) v_t(y) d\nu_t(y), \quad \forall x \in D_{t+1}, \tag{7}$$

where $v_t$ are output functions of the $t_{th}$ integral operator layer, $\kappa^{(t)} \in C(D_{t+1} \times D_t; \mathbb{R}^{d_{v_{t+1}} \times d_{v_t}})$. As shown in Fig. 3, depending on the choice of parameterization for the integral kernel and network architectures, two main integral kernel operator frameworks based on the graph kernel and the Fourier transform are proposed.

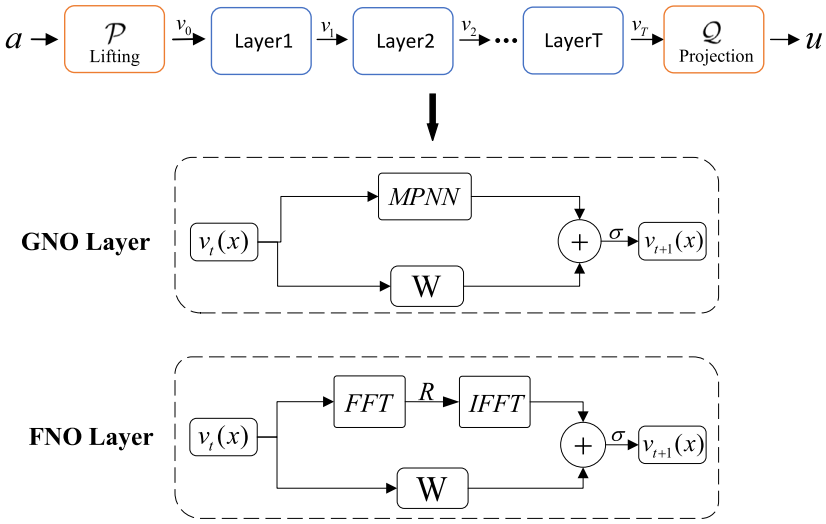



**Fig. 3.** Framework of integral kernel operators: The input function $a$ is fed into the point-by-point lifting operator $\mathcal{P}$, followed by the $T$-layer integration operator and the point-by-point nonlinear operation $\sigma$. Lastly, the point projection operator $\mathcal{Q}$ produces the function $u$.

Integral kernel operators are infinite-dimensional nonlinear operators, enabling the learning of nonlocal properties of $G$ through successive approximations with high accuracy. The architecture Eq. (6) is defined directly in the function space, rendering it independent of data discretization. GNO and FNO are primarily proposed by selecting different kernel functions in the operator layer. They will be introduced separately in the following sections. Moreover, the framework of integral kernel operators is general and can adopt various parameterization forms for the integral kernel functions to tackle other specialized application problems, in addition to the proposed FNO and GNO.

### 2.2.1. GNO

Graph neural networks (GNNs) have been widely employed in various fields since their introduction. For instance, Alet et al. [36] utilized GNNs to model Poisson's equation on different manifolds, constructing spatial graphs and leveraging nodes and edges for information transfer and aggregation. The integral kernel can learn directly through Message Passing Neural Networks (MPNNs) and Nyström's approximation techniques to approximate the solutions of the PDEs, which inspired the proposal of Graph Neural Operator (GNO) [20]. The GNO utilizes connections between graph neural networks to facilitate information transfer. Therefore, the integral kernel operator of GNO in Eq. (7) can be expressed as follows:

$$v_{t+1}(x) = \sigma\left(Wv_t(x) + \frac{1}{|N(x)|}\sum_{y\in N(x)}\kappa_\phi(e(x,y))v_t(y)\right), \tag{8}$$

where $W \in \mathcal{R}^{n\times n}$, $N(x)$ is the neighborhood of $x$ according to the graph, and $\kappa_\phi(e(x,y))$ is the neural network with the matrices in $\mathcal{R}^{n\times n}$ as the input edge features and output edge features.

However, when constructing graphs to approximate the solution of PDEs, long-range interactions are often overlooked due to the unfavorable scaling properties of GNO with respect to the number of nodes. Therefore, Li et al. [37] have proposed the Multipole Graph Neural Operator (MGNO), which is a framework that captures all-range interactions with linear complexity, enabling the learning of global properties of the operator. The MGNO decomposes the operator kernel into multi-level sub kernels to capture interactions ranging from short-range to long-range. This approach goes beyond capturing only neighboring interactions with linear complexity in terms of the number of nodes. The operator kernel of MGNO can be expressed as follows:

$$\begin{aligned} &K = K_{1,1} + K_{1,2}K_{2,2}K_{2,1} + K_{1,2}K_{2,3}K_{3,3}K_{3,2}K_{2,1} + \dots, \\ &K_{l',l} : v_l \mapsto v_{l'} = \int_{B\left(x,r_{l',l}\right)} \kappa_{l',l}(x,y)v_l(y)\mathrm{d}y, \end{aligned} \tag{9}$$

where $K_{1,1} = K_1$ represents the shortest range, $K_{1,2}K_{2,2}K_{2,1} \approx K_2$ represent the subsequent ranges, and so on. The central matrix $K_{l,l}$ is a $J_l \times J_l$ kernel matrix that corresponds to the discretization level $l$ described above. Additionally, the matrices $K_{l+1,l}, K_{l,l+1}$ are $J_{l+1}\times J_l$ and $J_l \times J_{l+1}$ wide and long respectively block transition matrices. Such a model captures better global information than the original graph neural operator and, therefore, has better generalization.

GNO is especially proficient in handling graph-formatted inputs since it does not depend on a regular mesh and can capture node interactions based on edge features. However, when confronted with large or dense datasets, GNO necessitates substantial memory resources and may exhibit slow execution. Moreover, augmenting the number of hidden layers in GNO can result in instability issues [17].

### 2.2.2. FNO

Despite previous proposals of neural operators like GNO, challenges related to computational cost and instability still persist. The Fourier transform is frequently employed in spectral methods for solving PDEs and holds significant importance in the field of deep learning [38,39]. Consequently, Li et al. [19] introduced an alternative neural operator architecture known as Fourier neural operator (FNO), which is defined directly in Fourier space. The FNO utilizes the Fast Fourier Transform (FFT) to compute the integral of the kernel in Eq. (7). By applying $\kappa(x,y) = \kappa(x-y)$ and the convolution theorem, the integral kernel operator of FNO in Eq. (7) can be expressed as:

$$(\mathcal{K}(a;\phi)v_t)(x) = \mathcal{F}^{-1}(R_\phi \cdot \mathcal{F}(v_t))(x), \quad \forall x \in D, \tag{10}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote Fourier transform and inverse Fourier transform respectively, $R_\phi$ is the Fourier transform of the periodic function $\kappa : D \to \mathbb{R}^{d_v\times d_v}$ parameterized by $\phi \in \Theta_\mathcal{K}$. Thus, from Eqs. (7) and (10), the single Fourier layer of the FNO is expressed in the following form:

$$v_{t+1} = \sigma\left(W_t v_t + \mathcal{F}^{-1}(R_t \cdot \mathcal{F}(v_t)) + b_t\right). \tag{11}$$

Importantly, unlike DeepONet and GNO, FNO relies on the FFT, which is defined only on uniformly discretized grids. As a result, FNO is grid-based and unable to overcome the "curse of dimensionality" [40]. However, by utilizing the Fourier transform, FNO efficiently extracts features of the input function in the frequency domain, leading to significant improvements in accuracy and computational speed compared to deep learning methods like U-Net [41] and GNO. Due to its quasi-linear time complexity and state-of-the-art approximation capability, FNO has gained widespread attention in practical applications. Numerous works are continuously emerging based on FNO.

For instance, FNO has been applied to climate forecasting [42] and super-resolution techniques incorporating FNO and attention mechanisms [24]. FNO's advantages in terms of high speed and accuracy are expected to facilitate its application in various fields in the future.

### 2.3. Transformer-based neural operator

The attention mechanism has found widespread application in various tasks within the field of natural language processing since its proposal by Bahdanau et al. [43] in 2014. Due to its ability to significantly enhance model performance and generalization by enabling a shift from a global to a focused focus, the attention mechanism has been a powerful tool in numerous machine-learning tasks. The original formulation of the attention mechanism is as follows:

$$\boldsymbol{z}_i = \sum_{j=1}^{n} \alpha_{ij} \boldsymbol{v}_j, \quad \alpha_{ij} = \frac{\exp\left(h\left(\boldsymbol{q}_i, \boldsymbol{k}_j\right)\right)}{\sum_{s=1}^{n} \exp\left(h\left(\boldsymbol{q}_i, \boldsymbol{k}_s\right)\right)}, \tag{12}$$

where $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ are the query vector, key vector, and value vector, respectively, $\alpha_{ij}$ is the weight of attention, and $\boldsymbol{z}_i$ is the weighted output.

Cao et al. [21] first introduced the linear transformer into operator learning by proposing the GK-Transformer. They highlight a striking similarity between scaled dot product attention and Fourier-type kernel integral transforms, arguing that the attention mechanism can be regarded as a form of learnable integral operator. Coincidentally, Kovachki et al. [28] also argued that the attention mechanism is a particular case of the neural operator layer and gave theoretical proof for this view. The GK-Transformer incorporates the attention mechanism within a hidden layer by utilizing a CNN as the feature extractor. It can be regarded as a numerical product form of the integral:

$$\begin{aligned} \text{Galerkin-type: } \left(\mathbf{z}^j\right)_i &= \sum_{l=1}^{d} \frac{\left(\mathbf{k}^l \cdot \mathbf{v}^j\right)}{n} \left(\mathbf{q}^l\right)_i \\ &\approx \sum_{l=1}^{d} \left( \int_{\Omega} \left(k_l(\xi) v_j(\xi)\right) d\xi \right) q_l\left(x_i\right). \end{aligned} \tag{13}$$

Integrating the attention mechanism into the neural operator dramatically improves the performance of the model for solving PDEs, especially in complex problem scenarios. However, the input ($y_{in}$) and output ($y_{out}$) features of GK-Transformer are restricted to the same mesh, thereby constraining its applicability in scenarios involving multi-input and multi-scale problems. Subsequently, a general neural operator transformer (GNOT) architecture has been proposed, capable of accommodating multiple input functions, irregular grids, and multi-scale problems with high flexibility, allowing input and output at any location. The architecture of the GNOT model, represented in Fig. 4, can be expressed as follows:

$$\begin{aligned} \mathbf{z}_t^c &= \mathbf{z}_{t-1} + \text{Cross-Attn}(\hat{f}_w(\mathbf{a}), f_w(x)), \\ \tilde{\mathbf{z}}_t &= Gate(x) \otimes \text{FNNs}(\mathbf{z}_t^c) + \mathbf{z}_t^c, \\ \mathbf{z}_t^s &= \tilde{\mathbf{z}}_t + \text{Self-Attn}(\tilde{\mathbf{z}}_t), \\ \mathbf{z}_t &= Gate(x) \otimes \text{FNNs}(\mathbf{z}_t^s) + \mathbf{z}_t^s, \end{aligned} \tag{14}$$

where $\mathbf{a}$ and $x$ represent input functions and query points, $f_w$ and $\hat{f}_w$ denotes the encoding of input data with features using MLPs and *Gate* denotes a gated network of query point coordinates to compute the weights of multiple FFNs. The GNOT is characterized by its ability to prioritize operator learning and its capacity to handle diverse types of operator operations with significant expressive power. It demonstrates broad applicability across various fields, including image processing, and can be employed for learning and processing different types of operators.

### 2.4. Discussion

DeepONets, integral kernel operators, and transformer-based neural operators are three distinct frameworks for neural operators, each possessing unique advantages and limitations. In this section, we will compare and discuss the three types of operators from their frameworks, design content, complexity, and various other aspects. The comparison summary is shown in Table 1.

In integral kernel operator frameworks, both GNO and MGNO are implemented with graph neural networks. GNO employs a straightforward and direct approach, Nyström approximation and domain truncation to approximate the operator. MGNO builds upon GNO by incorporating multi-scale graphs into its framework. FNO employs the FFT to compute the integral kernel operator, while DeepONet directly relies on the universal approximation theorem to approximate mappings in infinite-dimensional function spaces. Unlike GNO and MGNO, FNO and DeepONet do not require sampling, resulting in faster computation. FNO achieves good prediction results only on regular grid data, while DeepONet is grid-independent and provides greater flexibility. Moreover, FNO necessitates complete data fields for training, whereas DeepONet does not encounter this issue. Nonetheless, FNO generally outperforms DeepONet in terms of accuracy and speed when solving PDEs such as Darcy, Burgers', etc., and it possesses discretization invariance (i.e., FNO shares the same model parameters among different discretization of the underlying function spaces) that DeepONet lacks [28].

In contrast to the integral kernel operator and DeepONet, which were originally proposed for solving PDEs, the transformer was initially applied in Natural Language Processing (NLP) tasks and later adopted in operator learning due to its efficient feature capture capabilities. The transformer-based neural operators do not exhibit significant advantages over other NOs when solving simple PDEs but achieve higher accuracy in solving complex high-dimensional PDEs. Moreover, they excel in leveraging the growing volume of data, utilizing their larger model capacity to minimize errors, which is particularly advantageous when handling large datasets. However, it is essential to note that transformer-based operators, although effective, necessitate more time and a more significant number of parameters compared to other methods.

In terms of complexity, both GNO and MGNO involve sampling; thus, their complexity depends on the number of sampled nodes $J'$. They run slower due to the graph connection structure, especially the multilevel graph structure in MGNO, which demands more memory. FNO has a complexity of $O(J \log J)$, but due to the use of FFT and the absence of the kernel network $\kappa$, FNO runs faster. Unlike the integral kernel operator, the choice of NNs in DeepONet's trunk and branch networks is flexible, so the complexity needs to be analyzed based on the specific choice of NNs. Since the DeepONet framework is very flexible, its complexity needs to be determined according to the specific network architecture of trunk and branch networks. The complexity of the transformer primarily depends on the length of the input sequence and the size of the model's parameters. The complexity of the transformer-based neural operator in Table 1 corresponds to the simplest attention architecture.

## 3. Extentions of neural operator based on different directions

Although neural operators such as DeepONet, FNO, and GK-Transformer have been proposed to solve PDEs and have demonstrated remarkable accuracy, these classical operator architectures still face several challenges. For example, GNO can become unstable as the network depth increases [44]. Consequently, researchers have conducted further investigations on these neural operators to enhance and extend their capabilities. In this section, we will introduce advancements in neural operators from three different directions: operator basis, physical information, and complex systems.

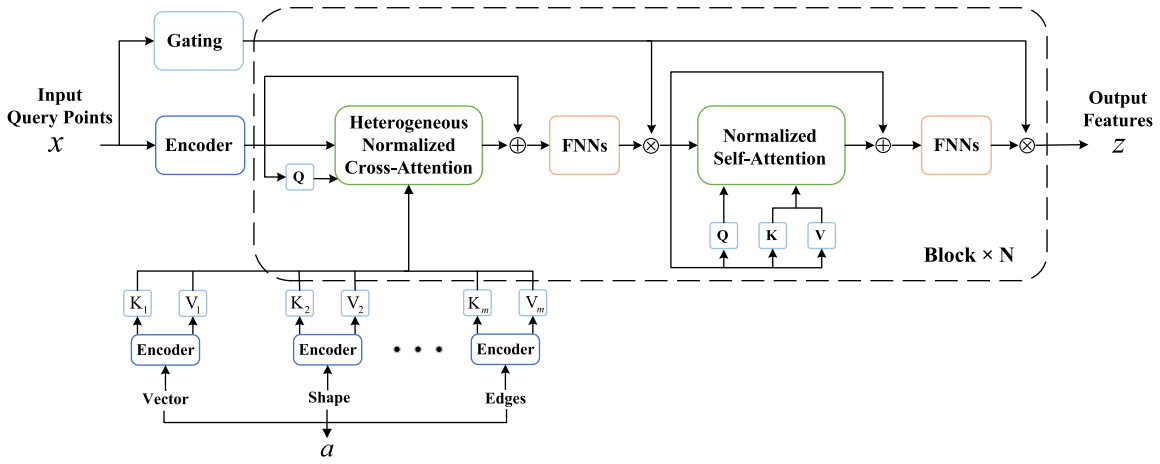



**Fig. 4.** Overview of the GNOT architecture: Initially, the input query points and $m$ input functions are encoded using distinct MLPs. Heterogeneous normalized cross-attention layers and normalized self-attention layers are employed to update the features of the query points. A gate network is employed, utilizing the geometric coordinates of the query points to compute the weighted average of multiple expert FFNs. Finally, the updated features are outputted.

**Table 1**
Comparison of FNO, GNO, MGNO, DeepONet and Transformer.

| | GNO | MGNO | FNO | DeepONet | Transformer |
|---|---|---|---|---|---|
| Framework | Nyström approxi-mation theorem | Multilevel diagram | Fourier convolution theory | Universal approxi -mation theorem | Attention mechanism |
| Graph-based | ✓ | ✓ | × | × | × |
| Kernel network | ✓ | ✓ | × | × | × |
| Complexity | $O(J'J)$ | $\sum_l O\left(J_l'^2 r_l^2\right) \sim O(J)$ | $O(JlogJ)$ | – | $O(J^2)$ |
| Parameter | Small | Small | Larger | Larger | Largest |
| Discretization -Invariance | ✓ | ✓ | ✓ | × | ✓ |

* $J$ is the number of uniformly sampled discretization points $\{x_1, \ldots, x_J\} \in D$, $J'$ is the number of randomly selected points.

### 3.1. Extension based on operator basis

Similar to FNO and GNO, DeepONet can also be represented in the form of a point-by-point parameterized kernel and a discretized integral operator [28], which motivates the proposal of new neural operator variants based on underlying basis functions. From Eq. (5), DeepONet can be expressed as follows:

$$G_\theta(a)(x) = \sum_{k=1}^{p} b_k(a) t_k(x) + b_0, \tag{15}$$

where $t_k$ can be regarded as the basis learned by DeepONet, and $b_k$ is the learned coefficient. The Fourier transform of a single Fourier layer in FNO is expressed as:

$$(\mathcal{F}v_t)(k) = \langle v_t, \psi(k)\rangle_{L(D)} \approx \sum_{x\in\mathcal{T}} v_t(x)\psi(x,k), \tag{16}$$

where $\psi(x,k) = e^{2i\pi\langle x,k\rangle} \in L(D)$ is the Fourier basis and $\mathcal{T}$ is the sampled grid. The choice of operator basis is one of the fundamental components of a neural operator. By selecting suitable basis functions, the model can capture patterns and structures in the data accurately. The properties of basis functions impact the overall characteristics of the neural operator. For instance, FNO attains superior computational efficiency compared to other operators through the utilization of Fourier basis functions. However, it is constrained to regular grids due to its reliance on FFT. To overcome the limitations of these inherent properties of existing operator basis, numerous researchers have proposed novel neural operators by modifying and optimizing the choice of operator basis.

#### 3.1.1. Extension based on operator basis function

*Wavelet Neural Operator (WNO)*: Wavelets have a rich history in signal processing [45]. Considering that the basis functions in the FFT of FNO are usually frequency localized and have no spatial resolution, the wavelet is both spatially local and frequency local. Gupta et al. [46] proposed the Multiwavelet (MWT) neural operator by introducing the wavelet transform into operator learning for the first time. The MWT neural operator utilizing a multiwavelet basis to replace the Fourier basis in the operator layer effectively enhances the accuracy of the operator solution and demonstrates robustness to the input signal. However, the MWT imposes specific requirements on the dataset, which only supports a resolution of the second power. Therefore, Tripura et al. [17] proposed the Wavelet Neural Operator (WNO), which employs wavelet basis functions for learning the kernel $\mathcal{K}_\phi$ in the wavelet domain. Let $\mathcal{W}$ and $\mathcal{W}^{-1}$ denote the forward and inverse wavelet transforms, respectively. The transformation is described by the following integral equation:

$$\begin{aligned}(\mathcal{W}\Gamma)_j(\alpha,\beta) &= \int_D \Gamma(x)\frac{1}{|\alpha|^{1/2}}\psi\left(\frac{x-\beta}{\alpha}\right)dx,\\ \left(\mathcal{W}^{-1}\Gamma_w\right)_j(x) &= \frac{1}{C_\psi}\int_0^\infty\int_D\left(\Gamma_w\right)_j(\alpha,\beta)\frac{1}{|\alpha|^{1/2}}\tilde{\psi}\left(\frac{x-\beta}{\alpha}\right)d\beta\frac{d\alpha}{\alpha^2},\end{aligned} \tag{17}$$

where $(\Gamma_w)_j(\alpha,\beta) = (\mathcal{W}\Gamma)_j(\alpha,\beta)\varphi((x-\beta)\alpha) \in \mathcal{L}^2(\mathcal{R})$ is a scaled and translated mother wavelet, usually called a sub-wavelet. WNO directly parameterizes the kernel function $k_\phi$ in the wavelet space; the kernel integral can be expressed as:

$$\left(\mathcal{K}(\phi) * v_t\right)(x) = \mathcal{W}^{-1}\left(R_\phi \cdot \mathcal{W}\left(v_t\right)\right)(x), \quad x \in D, \tag{18}$$

where $R_\phi = \mathcal{W}(k_\phi)$ represents the wavelet transform of the kernel function $k_\phi$. The wavelet basis offers multi-scale analysis capabilities compared to the Fourier basis. Moreover, wavelets exhibit spatial and frequency localization, effectively handling signals with discontinuities and spikes. In contrast, Fourier basis functions are global, and analyzing the entire signal may not adequately capture local features. Furthermore, by employing wavelet basis functions instead of Fourier bases,

WNO overcomes the limitation of FNO being restricted to a regular grid.

By leveraging the advantages of the wavelet transform, the DeepONet network architecture is expanded to include the wavelet, resulting in the Wavelet DeepONet (W-DeepONet) [47]. W-DeepONet consists of two branch networks and two trunk networks. The inputs to the branch and trunk networks are the corresponding wavelet coefficients $\boldsymbol{\eta}_i$ and time coordinates $\boldsymbol{x}_i$, yielding the approximation wavelet coefficients $\boldsymbol{\eta}_{i,A}$, $\boldsymbol{x}_A$, and detail wavelet coefficients $\boldsymbol{\eta}_{i,D}$, $\boldsymbol{x}_D$. Therefore, the solution operator at coordinates $\boldsymbol{x}_A$ and $\boldsymbol{x}_D$ is expressed as follows:

$$\begin{aligned} G_\theta\left(\eta_{i,A}\right)\left(x_A\right) &= \sum_{k=1}^{q/2} b_k\left(\eta_{i,A}\left(t_1\right), \eta_{i,A}\left(t_2\right), \ldots, \eta_{i,A}\left(t_{n_{sen}}\right)\right) \cdot t_k\left(x_A\right), \\ G_\theta\left(\eta_{i,D}\right)\left(x_D\right) &= \sum_{k=q/2+1}^{q} b_k\left(\eta_{i,D}\left(t_1\right), \eta_{i,D}\left(t_2\right), \ldots, \eta_{i,D}\left(t_{n_{sen}}\right)\right) \cdot t_k\left(x_D\right). \end{aligned} \tag{19}$$

The inverse wavelet transform is performed after obtaining $G_\theta$ to obtain the predicted response in the time domain. W-DeepONet incorporates temporal coordinate history information into the base, thereby enhancing the performance of the operator and enabling the capture of both transient and non-transient responses.

*Laplace Neural Operator(LNO):* Due to the limitation of FNO in capturing transient responses and non-periodic signals, Cao et al. [48] proposed the Laplace Neural Operator with a Pole-residue formulation (PR-LNO). Similar to the concept of WNO, PR-LNO employs the Laplace transform instead of FFT to learn network parameters. It uses the Pole-residue formulation in the Laplace layer to establish the relationship between the Laplace transform of the input function and the Laplace transform of the output function, i.e., replacing the kernel integral operator with the operator defined in the Laplace domain :

$$U(s) = \mathscr{L}\{(\kappa(\mathbf{f};\phi) * v)(t)\} = K_\phi(s)V(s), \tag{20}$$

where $\mathscr{L}\{.\}$ and $\mathscr{L}^{-1}\{.\}$ are the Laplace transform and the inverse Laplace transform, $K_\phi(s) = \mathscr{L}\left\{\kappa_\phi(t)\right\}$ and $V(s) = \mathscr{L}\{v(t)\}$. Use the Pole-residue form to express $K_\phi(s)$, that is

$K_\phi(s) = \sum_{n=1}^{N} \frac{\beta_n}{s-\mu_n}$,

Finally, after a series of transformations, we can get:

$$v_t = \sum_{n=1}^{N} \gamma_n \exp\left(\mu_n t\right) + \sum_{\ell=-\infty}^{\infty} \lambda_\ell \exp\left(i\omega_\ell t\right). \tag{21}$$

The first summation term on the right side of the equal sign represents the transient response corresponding to the system poles, while the second summation term represents the steady-state response in the frequency domain. PR-LNO outperforms FNO in capturing undamped transient responses (FNO is unsuitable for analyzing unstable systems). Moreover, certain functions lack a Fourier transform (such as $|x(t)|$) due to non-integrability, whereas PR-LNO tackles these issues. Furthermore, the Laplace transform can be extended to Riemannian manifolds, allowing its application to complex geometries while preserving discretization invariance [49].

*Spectral Neural Operator(SNO):* The output function of the vanilla FNO and DeepONet is parameterized by the neural network, resulting in opaque outputs. Furthermore, the nonlinear activation of FNO introduces higher frequencies, thereby distorting lower frequencies as well. In the spectral method [50], trigonometric polynomials and Chebyshev polynomials are commonly employed for solving PDEs. Consequently, the Spectral Neural Operator (SNO) [51], which employs a finite series representation $\sum c_n f_n(x)$ of a function, is proposed:

$$\sum_i b_i f_i(x) = a_{\text{in}}\ (x) \xrightarrow{\text{N}} u_{\text{out}}\ (x) = \sum_i d_i f_i(x), \tag{22}$$

where $f_i(x)$ are Chebyshev polynomials or complex exponential. The neural network maps the finite number of input coefficients $b_i$ to the finite number of output coefficients $d_i$. SNO enables lossless operations on functions and achieves interpolation on fine grids with arbitrary precision. However, in its current version, SNO can only handle smooth input and output data due to the Gibbs phenomenon [52]. Furthermore, SNO's currently utilized basis functions are completely non-adaptive, which hinders its performance in solving complex problems. Liu et al. proposed the Orthogonal Polynomial Neural Operator (OPNO) [53] based on the spectral method, utilizing a compact combination of Chebyshev polynomials as the polynomial basis, which automatically satisfies specific boundary conditions:

$$v_{t+1} = v_t + \sigma\left(W v_t + b_t + S^{-1}\left(A_t \cdot C_h\left(v_t\right)\right)\right), \tag{23}$$

where $C_h$ is the fast Chebyshev transform, $A_t$ is a quasi-diagonal matrix with a bandwidth of w and $S$ is the Shen transform. or to $d$-dimensional cases. In order to construct the fast Shen transform $S$ that maps an arbitrary function $u \in C(I)$ to its expansion coefficients on the compact combination basis, they devised $S = C_p \circ C_h$ inspired by the compact combination basis of Chebyshev polynomials $\{\phi_k(x)\}$: the fast Chebyshev transform $C_h$ that maps the function into its Chebyshev coefficients is taken first $(x \to T_k()$, and then the fast compacting transform $C_p$ mapping the Chebyshev coefficients to the coefficients of compact combinations $\phi_k\left(T_k \to \phi_k\right)$ follows. OPNO is employed to solve partial differential equations with Dirichlet, Neumann, and Robin boundary conditions.

Similarly, the Spectral Coefficient Learning via Operator Network (SCLON) [54] was proposed based on the spectral method. SCLON uses orthogonal function expansion and adopts linear combinations of Legendre polynomials as basis functions:

$$\phi_n(x) = L_n(x) + a_n L_{n+1}(x) + b_n L_{n+2}(x), \tag{24}$$

where $L_n$ represents the Legendre polynomial of order $n$. Neural operators based on spectral methods improve the efficiency and transparency of solving PDEs by combining the advantages of spectral methods, such as high accuracy, high efficiency, and generalization, with the capabilities of deep neural networks. However, they are often limited to solving some specific types of PDEs.

*Nonlocal Neural Operator(NNO):* Nonlocal operators have been utilized in diverse fields [55–57], demonstrating their effectiveness in representing long-range dependencies and resolution invariance. Furthermore, rigorous identifiability analysis and convergence studies have been proposed [58]. In the previous analysis of FNO's generalized operator approximation theorem [59], the basic form was restricted to problems with periodic geometry. Subsequently, Lanthaler et al. [60] introduced a novel general operator approximator known as Nonlocal Neural Operator (NNO), enabling operator approximation between function spaces defined on arbitrary geometric shapes,

$$(\mathcal{K}_\ell v)(x) = \sum_{m=0}^{M} \left\langle T_{\ell,m} v, \psi_{\ell,m} \right\rangle_{L^2\left(\Omega;\mathbb{R}^{d_c}\right)} \phi_{\ell,m}(x), \tag{25}$$

where $T_{\ell,m}$ is a learnable parameter. The nonlocal operator basis can be chosen according to the situation. In the simplest case, when the chosen basis function is the average of the input function, it is referred to as ANO. ANO exhibits nonlinearity with average nonlocality and applies to domains with arbitrary geometry. Additionally, ANO serves as an analytical tool for deriving novel theoretical results regarding neural operator architectures such as FNO, PR-LNO, WNO, etc. [60]. In fact, if the nonlocal operator basis function is selected as the Fourier basis function in periodic geometry, the operator simplifies to FNO. Building upon the advantages of nonlocal operators, You et al. [44] proposed the Nonlocal Kernel Network (NKN) by integrating GNO with NNO, resulting in improved stability and generalization.

*3.1.2. Operator basis extension based on decomposition*

*Factorized-FNO(F-FNO):* FNO's performance significantly deteriorates with complex geometries and noisy data, and there is also a tendency for FNO to become unstable with increasing network depth. To address these issues, the Factorized Fourier Neural Operator (F-FNO) was introduced [61]. F-FNO incorporates the Fourier factor decomposition, where each spatial dimension is independently processed in Fourier space. Unlike the Eq. (10) of FNO, this decomposition is expressed as follows:

$$\mathcal{F}^{(\ell)}\left(v^{(\ell)}\right)=\sum_{d\in D}\left[\mathrm{IFFT}\left(R_d^{(\ell)}\cdot\mathrm{FFT}_d\left(v^{(\ell)}\right)\right)\right], \tag{26}$$

where $\ell$ is number of lagers. Fourier factor decomposition ensures the rapid convergence of the Fourier series [62] and enables independent learning of the characteristics of each dimension in Fourier space. Consequently, compared to FNO, F-FNO can reduce model complexity by an order of magnitude and tackle higher dimensional problems.

*POD-DeepONet (POD-DON):* Empirical Modal Decomposition (POD) retrieves a set of orthogonal basis functions (known as POD modes) by employing singular value decomposition or eigenvalue decomposition on the data. This technique effectively extracts the dominant modes from extensive data, thereby reducing dimensionality and capturing crucial features. Building upon the concept of utilizing the POD representation function [63], Lu et al. [64] introduced the POD-DeepONet. This approach employs the POD basis as the input to the trunk network, effectively emulating the prior learning of basis functions through POD, while utilizing a branch network to learn the coefficients associated with the POD basis. The output of POD-DeepONet is represented as follows:

$$G_\theta(a)(x)=\sum_{k=1}^{p}b_k(a)\phi_k(x)+\phi_0(x), \tag{27}$$

where $\phi_0(x)$ is the average function of $u(x)$ calculated from the training dataset. $\{b_1,b_2,\ldots,b_p\}$ is the output of the branch network, $\{\phi_1,\phi_2,\ldots,\phi_p\}$ is the precomputed POD modes of $u(x)$. By capturing the primary change patterns in the data and reducing its dimensionality to a certain extent, these POD basis effectively enhance the performance of DeepONet.

*Spatio-Spectral Neural Operator(SSNO):* Recognizing that purely spatial methods solely capture instantaneous spatial information, Muhammad et al. [65] introduced the spatio-spectral Neural Operator (SSNO). SSNO incorporates an input mixer into the Fourier convolution layer to prevent the loss of information and decomposes $u_i$ and $u_f$ using low-pass filters to capture the low-frequency features:

$$\begin{aligned}&u_f=\alpha u_0+(1-\alpha)*u_t,\\&u_{t+1}=\sigma(u_f*K_f+u_t*K),\end{aligned} \tag{28}$$

where $\alpha$ is the weight, $K_f$ is the Fourier convolution block, $K$ is spatial convolution with filters, $\sigma$ is Gelu activation function. By including feature information, the network maintains complete input information at all times, leading to a significant enhancement in the performance of the neural operators.

*Koopman neural operator(KNO):* Xiong et al. proposed the Koopman Neural Operator (KNO) [66] as a solution to the challenge of predicting long-term dynamics of PDEs. KNO utilizes the Fourier transform mapping to map features into Fourier space. In this space, the spatial basis functions spanned by the Hankel matrix are used to approximate the invariant subspace of the target Koopman operator. The input is decomposed into high-frequency and low-frequency modal components through truncated Fourier transform for learning. The original Koopman operator is approximated using the Hankel matrix according to the following procedure:

$$\hat{\gamma}_{t'}=[\mathbf{g}^{-1}(1-\lambda)\mathcal{F}^{-1}\circ\overline{\mathcal{K}}_\varepsilon^r\circ\mathcal{F}\circ\mathbf{g}\left(\hat{\gamma}_{[t-m\varepsilon,t]}\right)+\lambda C\circ\mathbf{g}\left(\hat{\gamma}_{[t-m\varepsilon,t]}\right)]^\top(m), \tag{29}$$

where $\mathcal{F}$ is the Fourier transform, $\mathbf{g}$ is an observation function (which is a single non-linear layer with $tanh(\cdot)$ activation function). And $\hat{\gamma}_{[t-m\varepsilon,t]}$ is a vector $\left[\hat{\gamma}_{t-m\varepsilon},\ldots,\hat{\gamma}_t\right]$ defined by $m\in\mathbb{N}$, the dimension of delay-embedding. $\widehat{\mathcal{K}}_t^t:G\left(\mathbb{R}^{d_\gamma}\times T\right)\to\mathbb{K}$ for any $t\in T$, which is the original Koopman operator to $\mathbb{K}$.

*3.1.3. Extension based on basis space*

*Improved-DeepONet:* With the advantage of high-dimensional feature space, which highlights the main features of the data and removes redundant information, Wang et al. [67] proposed an Improved-DeepONet. This model embeds the DeepONet's input $a$ and $x$ into the high-dimensional feature space through two encoders $A$ and $B$ respectively:

$$\begin{aligned}&\boldsymbol{A}=\sigma\left(\boldsymbol{W}_a\boldsymbol{a}+\boldsymbol{b}_a\right),\quad \boldsymbol{B}=\sigma\left(\boldsymbol{W}_x\boldsymbol{x}+\boldsymbol{b}_x\right),\\&\boldsymbol{H}_a^{(1)}=\sigma\left(\boldsymbol{W}_a^{(1)}u+\boldsymbol{b}_a^{(1)}\right),\quad \boldsymbol{H}_x^{(1)}=\sigma\left(\boldsymbol{W}_x^{(1)}\boldsymbol{x}+\boldsymbol{b}_x^{(1)}\right),\\&\boldsymbol{Z}_a^{(l)}=\sigma\left(\boldsymbol{W}_a^{(l)}\boldsymbol{H}_a^{(l)}+\boldsymbol{b}_a^{(l)}\right),\quad \boldsymbol{Z}_x^{(l)}=\sigma\left(\boldsymbol{W}_x^{(l)}\boldsymbol{H}_x^{(l)}+\boldsymbol{b}_x^{(l)}\right),\\&\quad l=1,2,\ldots,L-1,\\&\boldsymbol{H}_a^{(l+1)}=\left(1-\boldsymbol{Z}_a^{(l)}\right)\odot\boldsymbol{A}+\boldsymbol{Z}_a^{(l)}\odot\boldsymbol{B},\quad l=1,\ldots,L-1,\\&\boldsymbol{H}_x^{(l+1)}=\left(1-\boldsymbol{Z}_x^{(l)}\right)\odot\boldsymbol{A}+\boldsymbol{Z}_x^{(l)}\odot\boldsymbol{B},\quad l=1,\ldots,L-1,\\&\boldsymbol{H}_a^{(L)}=\sigma\left(\boldsymbol{W}_a^{(L)}\boldsymbol{H}_a^{(L-1)}+\boldsymbol{b}_a^{(L)}\right),\quad \boldsymbol{H}_x^{(L)}=\sigma\left(\boldsymbol{W}_x^{(L)}\boldsymbol{H}_x^{(L-1)}+\boldsymbol{b}_x^{(L)}\right),\\&G_\theta(\boldsymbol{a})(\boldsymbol{x})=\left\langle\boldsymbol{H}_a^{(L)},\boldsymbol{H}_x^{(L)}\right\rangle,\end{aligned} \tag{30}$$

where $\left\{\boldsymbol{W}_a^{(l)},\boldsymbol{b}_a^{(l+1)}\right\}_{l=1}^{L+1}$ and $\left\{\boldsymbol{W}_x^{(l)},\boldsymbol{b}_x^{(l+1)}\right\}_{l=1}^{L+1}$ are the weights and biases of the branch and trunk networks, $\theta$ is trainable parameter, $\sigma$ denotes a activation function. The embedding of input coding into a high-dimensional feature space and the utilization of point-by-point multiplication at each layer of the network facilitates the merging of information. This not only aids in the propagation of the input signal through the DeepONet but also significantly enhances its ability to represent nonlinearities due to the extensive use of point-by-point multiplication.

*Latent Spectral Models (LSM):* The vanilla DeepONet, GNO, and GK-Transformer employ a high-dimensional spatial coordinate system for solving tasks, while FNO operates in the spectral domain. Nonetheless, the curse of dimensionality phenomenon [68] suggests that conducting the solution process in a high-dimensional space incurs significant computational costs. In light of this, Wu et al. [69] introduced the Latent Spectral Model (LSM), which utilizes the attention mechanism to project high-dimensional data into a compact latent space. The LSM employs the triangular basis function as the basis function, represented as follows:

$$\mathbf{T}_{\boldsymbol{y}}=\mathbf{T}_{\boldsymbol{x}}+\mathbf{w}_0+\mathbf{w}_{\sin}\begin{bmatrix}\sin\left(\mathbf{T}_{\boldsymbol{x}}\right)\\\vdots\\\sin\left(\frac{N}{2}\mathbf{T}_{\boldsymbol{x}}\right)\end{bmatrix}+\mathbf{w}_{\cos}\begin{bmatrix}\cos\left(\mathbf{T}_{\boldsymbol{x}}\right)\\\vdots\\\cos\left(\frac{N}{2}\mathbf{T}_{\boldsymbol{x}}\right)\end{bmatrix}, \tag{31}$$

where $\mathbf{w}_0\in\mathbb{R}^{d_{\text{latent}}},\mathbf{w}_{\sin}\in\mathbb{R}^{1\times\frac{N}{2}},\mathbf{w}_{\cos}\in\mathbb{R}^{1\times\frac{N}{2}}$ is the learnable parameter, and $\mathbf{T}_{\boldsymbol{x}}$ is the input hidden token. LSM learns the operator in the latent space by transforming, which efficiently removes the redundancy information. This enables LSM to accurately capture the intrinsic physical information, providing a notable accuracy advantage when dealing with vast high-dimensional data.

*L-DeepONet (L-DON):* Similar to LSM, Kontolati et al. [70] incorporated the latent space into DeepONet, resulting in the development of L-DeepONet. L-DeepONet enhances the accuracy and generalization capabilities of the model by utilizing an unsupervised autoencoder to project the data into the latent space. The trunk network then takes the temporal coordinates of the PDEs' output $\zeta=\left\{t_i\right\}_{i=1}^{T}$ as input,

$$\begin{aligned}&\mathcal{J}_{\theta_{\text{encoder}}}:\{\mathbf{a},\mathbf{u}\}\equiv\mathbf{z}\mapsto\{\mathbf{a}^r,\mathbf{u}^r\}\equiv\mathbf{z}^r,\\&G_\theta\left(\mathbf{a}_1^r\right)(\zeta)=\textstyle\sum_{i=1}^{p}b_i\left(\mathbf{a}_1^r\left(x_1\right),\mathbf{a}_1^r\left(x_2\right),\ldots,\mathbf{a}_1^r\left(x_d\right)\right)\cdot\mathrm{tr}_i(\zeta),\\&\mathcal{J}_{\theta_{\text{decoder}}}:\{\mathbf{a}^r,\mathbf{u}^r\}\equiv\mathbf{z}^r\mapsto\{\tilde{\mathbf{a}},\tilde{\mathbf{u}}\}\equiv\tilde{\mathbf{z}},\end{aligned} \tag{32}$$

where $[b_1,b_2,\ldots,b_p]$ is the branch network output and $[tr_1,tr_2,\ldots,tr_p]$ is the trunk network output. By serving as a basis space, the latent space effectively eliminates redundant information, facilitating the capture of feature information. Consequently, L-DeepONet surpasses classical DeepONet in terms of accuracy and computational efficiency when dealing with various time-dependent PDEs. However, due to its retention of only the temporal basis while convolving the spatial dimension, L-DeepONet is unsuitable for spatial interpolation (i.e., extrapolation).

Moreover, the neural operator that introduces latent space is employed to accelerate the prediction of phase field-based microstructure evolution problems, resulting in enhanced accuracy and speed [71].

*En-DeepONet:* The Eikonal equation plays a vital role in earthquake source detection [72], but DeepONet exhibits weakness in learning operators whose output solutions shift spatially, corresponding to changes in the input function. Haghighat et al. proposed the Enriched DeepONet (En-DeepONet) [73], which introduces addition and subtraction operations to the combination of the outputs of the branch and trunk networks, in addition to multiplication operations:

$$\begin{aligned} G(a)(x) \approx & \sum_{k=1}^{p} w_k^{+} \left(t_k(x) + b_k\left(a_1, a_2, \ldots, a_p\right)\right) \\ & + \sum_{k=1}^{p} w_k^{-} \left(t_k(x) - b_k\left(a_1, a_2, \ldots, a_p\right)\right) \\ & + \sum_{k=1}^{p} w_k^{*} \left(t_k(x) \cdot b_k\left(a_1, a_2, \ldots, a_p\right)\right), \end{aligned} \tag{33}$$

where $w_k$ are summation weights (linear layer). Incorporating addition and subtraction operations into the integration of branch and trunk networks facilitates the acquisition of broader mathematical operators.

*Nonlinear Manifold Decoders (NOMAD):* Classical neural operators like DeepONet use linear decoders to approximate target functions in finite-dimensional linear subspaces, but this can be inefficient when the data lies on low-dimensional nonlinear manifolds—known as the Operator Learning Manifold Hypothesis. To overcome this, the NOMAD [74] framework employs a nonlinear decoder that better captures the intrinsic geometry of functional data. Specifically, the decoder is defined as:

$$G_\theta(a)(x) = f_{dec}(b_1(a), \ldots, b_p(a), t_1(x), \ldots, t_p(x)), \tag{34}$$

here, $f_{dec}$ is a neural network, which can be implemented using architectures such as feedforward neural networks (FNNs) or convolutional neural networks (CNNs). Unlike Eq. (15), which represents the original DeepONet formulation, NOMAD expands the basis space from purely coordinate features to the joint feature space of both the input function $a$ and the spatial coordinate $x$. This richer representation enables more efficient dimensionality reduction, improved approximation accuracy, and reduced computational cost compared to traditional linear-decoder-based methods.

*Waveformer:* While existing neural operator frameworks perform well in learning solution operators, they often fail to capture long-term temporal dependencies. To address this, the Waveformer [75] architecture integrates wavelet transforms for spatial and frequency localization with transformers to model long-range temporal dynamics.

$$v_{t+1} = G_t\left(v_t\right) = \mathcal{W}^{-1}\left(T_{\mathcal{W}}\left(\mathcal{W}\left(v_t\right)\right)\right) + T_R\left(v_t\right), \tag{35}$$

where $\mathcal{W}(\cdot)$ and $\mathcal{W}^{-1}(\cdot)$ denote the forward and inverse wavelet transforms, $T_{\mathcal{W}}$ denotes the transformer acts on the wavelet domain. Moreover, the action of transformer $T_R$ yields a nonlinear transformation on the physical space, which in effect replaces the series of activations and linear transformations $W_t$ in Eq. (11). Waveformer extracts features from both the wavelet (frequency) domain and the physical (spatial) domain via dedicated transformer modules, and then fuses these representations. By leveraging the multi-resolution nature of wavelets and the global context modeling ability of transformers, Waveformer effectively captures complex multi-scale spatial patterns along with long-term temporal evolution.

### 3.1.4. Experiments

In this section, to compare and analyze the applicability of the neural operators with different basis extensions, we select eight datasets with different dimensionality, continuity, and regularity (The relative L2 norm error (%) for all benchmarks is presented in Table 2, respectively. The reported errors represent the mean ± 1 s.d. of three replicate experiments). The details of PDE datasets are in Appendix A.1. We use the $L_2$ relative errors predicted on dataset instances with different features to evaluate the performance of the framework. The parameter settings can be seen in Appendix B, and numerical experiments are performed on a single RTX 4090 GPU.

**Table 2**
Summary of experiment benchmarks.

| | Physics | Geometry | Dim | Feature |
|---|---|---|---|---|
| Burgers'BC | Fluid; gas | Regular grid | 1D | Periodicity; nonlinearity |
| Burgers' discontinuity | Fluid | Regular grid | 2D | Discontinuity |
| Advection | Fluid | Regular grid | 2D | Discontinuity; linearity |
| Darcy | Fluid | Regular grid | 2D | Boundary |
| Navier–Stokes | Fluid | Regular grid | 3D | Viscosity; nonlinearity |
| Allen-Cahn | Fluid; gas | Regular grid | 2D | Transient; nonlinearity |
| Airfoil | Gas | Structured mesh | 2D | Complex boundary |
| Elasticity | Solid | Structured mesh | 2D | Complex boundary |

*Discussions:* As shown in Tables 3 and 4, we conducted stability tests on FNO and F-FNO using network architectures of 4-layer and 24-layer. It is evident that as the number of network layers increases, FNO becomes unstable on most benchmark datasets, particularly on non-smooth PDEs such as Advection, Burgers' discontinuity, and Allen-Cahn equation, where it completely fails to converge. In contrast, since the Fourier factorization and improved residual connections proposed in F-FNO reduce the model complexity and allow it to scale to deeper networks, it achieves the best performance on the two benchmarks while maintaining stability and convergence. WNO effectively captures spatial features, enabling it to achieve excellent accuracy across most benchmark datasets. Furthermore, compared to FNO, it demonstrates superior accuracy when applied to irregular data. Due to the Gibbs phenomenon, SNO only applies to smooth input and output data. Additionally, SNO struggles to perform well on complex benchmark datasets using a non-adaptive basis function. LNO is designed to address specific classes of PDEs. It exhibits limited accuracy when applied to Darcy flow equations but demonstrates strong performance in solving Burgers equations, particularly those exhibiting discontinuous characteristics. With the introduction of the attention mechanism, LSM showcases optimal or near-optimal performance on most benchmark datasets, demonstrating its versatility in handling different PDEs. Moreover, LSM overcomes the limitations of FNO on regular grids by utilizing a triangular basis. As in Tables 3 and 4, DeepONet is unsuitable for solving PDEs with transient responses, such as the Allen-Cahn equation. By leveraging POD to enhance feature capture capabilities, POD-DeepONet achieves better performance than DeepONet on most benchmark datasets. L-DeepONet utilizes an unsupervised autoencoder to project data into the latent space, resulting in improved performance on datasets with low dimensionality. However, its performance diminishes when solving high-dimensional complex problems such as the Navier–Stokes equation, as it only retains the temporal basis, limiting its applicability in solving PDEs with significant spatial variations. And Waveformer exhibits significant advantages in addressing time-dependent PDEs, achieving optimal or near-optimal accuracy when solving time-dependent discontinuous Burgers and Navier–Stokes equations. This is attributed to its effective combination of frequency and spatial domains, as well as the incorporation of the Transformer architecture, which enhances its ability to capture long-range dependencies in the data. In most experiments, NOMAD did not outperform DeepONet. Although it introduces a nonlinear manifold decoder from the perspective of the Operator Learning Manifold Hypothesis, this design compromises the architecture of DeepONet, which is based on the universal approximation theorem of operators.

The numerical experiment results confirm the advantageous properties of different operators. For example, WNO utilizes the wavelet basis instead of the Fourier basis, enabling the capture of local features in

**Table 3**
In all benchmark tests, we compare the performance of eight operator methods by recording the test set $L_2$ relative error. A smaller $L_2$ relative error indicates better performance. For clarity, the best results are shown in bold, and the second-best results are underlined. We only compare L-DeepONet [70] and Waveformer [75] on the Burgers' discontinuity equation since it is specifically designed for time-dependent PDEs.

| | Burgers-BC (%) | Burgers- discontinuity (%) | Darcy (%) | AC_BC2d (%) | Airfoil (%) | Elasticity (%) |
|---|---|---|---|---|---|---|
| FNO | **0.06±0.00** | 0.21±0.03 | 1.35±0.03 | 3.74±0.27 | 6.41±6.65 | 22.01±0.00 |
| FNO(24) | 4.74±1.67 | 100±0.00 | 25.28±0.00 | 99.98±0.00 | 6.11±2.76 | 18.13±0.00 |
| DeepONet | 2.15±0.09 | 1.34±0.05 | 2.98±0.03 | 54.93±0.31 | 10.67±0.12 | 41.37±1.20 |
| WNO | 0.98±0.16 | 0.38±0.18 | 1.56±0.02 | 4.32±0.04 | 3.03±0.00 | 8.15±0.26 |
| SNO | 10.94±0.46 | 14.05±0.66 | 25.56±0.00 | 99.98±0.00 | 10.05±0.28 | 100.96±0.32 |
| LNO | 8.57±0.87 | 1.65±0.26 | 3.37±0.04 | 13.82±0.26 | 1.94±0.07 | 55.53±0.00 |
| F-FNO(4) | 0.51±0.1 | 0.32±0v | 0.75±0.02 | 1.22±0.23 | 4.18±0.58 | 37.09±0.26 |
| F-FNO(24) | 0.36±0.01 | 2.89±4.47 | **0.52±0.01** | **1.07±0.06** | 2.9±0.16 | 36.42±0.34 |
| POD-DeepONet | 1.94±0.07 | 0.15±0 | 2.32±0.03 | 55.93±0.81 | 10.5±0.00 | 30.23±0.21 |
| NOMAD | 3.23±0.16 | 0.37±0.09 | 12.33±2.18 | – | 5.22±1 | 53.21±1.42 |
| LSM | 2.45±0.09 | 0.19±0.17 | 0.69±0.01 | 3.09±0.05 | **0.62±0.05** | **2.42±0.22** |
| L-DeepONet | – | 4.75±0.14 | – | – | – | – |
| Waveformer | – | **0.12±0.02** | – | – | – | – |

**Table 4**
We compare the performance of eight operator methods by recording the test set $L_2$ relative error by two different training approaches in time. A smaller $L_2$ relative error indicates better performance. The best results are shown in bold, and the second-best results are shown underlined.

| | advection1d_time (%) | advection2d (%) | NS2d_time (%) | NS3d (%) |
|---|---|---|---|---|
| FNO | 10.36±0.77 | **0.22±0.01** | 0.62±0.06 | **0.31±0.02** |
| FNO(24) | 73.86±20.28 | 80.28±0.00 | 100±0.00 | 30.17±0.00 |
| DeepONet | – | 0.32±0.04 | – | 1.78±0.02 |
| WNO | **1.08±0.11** | 0.89±0.03 | 3.8±0.04 | 0.42±0.01 |
| SNO | – | 60.62±0.00 | – | – |
| LNO | 80.41±0.01 | 26.39±0.22 | 2.65±0.09 | 5.64±0.24 |
| F-FNO(4) | 28.76±4.62 | 3.02±1.31 | 0.93±0.01 | 0.71±0.01 |
| F-FNO(24) | 79.64±0.52 | 1.05±0.35 | 0.7±0.03 | 0.56±0.01 |
| POD-DeepONet | – | 0.4±0.00 | – | 1.36±0.03 |
| NOMAD | – | 1.15±0.04 | – | 3.43±0.11 |
| LSM | 94.96±5.76 | 0.68±0.03 | **0.5±0.01** | 2.16±0.02 |
| L-DeepONet | – | 5.48±0.11 | – | 25.66±0.48 |
| Waveformer | – | – | 0.89±0.01 | – |

the spatial domain. So, it achieves higher accuracy compared to benchmark methods when solving discontinuous Advection equations with time-stepped iteration. Similarly, F-FNO and POD-DeepONet employ decomposition techniques to capture additional feature information, leading to comparable or even superior accuracy and enhanced robustness compared to the benchmark methods (FNO and DeepONet). This emphasizes that the properties of neural operators can be attributed to the nature of the operator basis, which exhibits distinct characteristics based on the selection of the basis functions. It means that the selection and improvement of the operator basis can serve as a starting point for improving neural operators.

Finally, to test the performance of neural operators in solving time-dependent PDEs, we employed two different approaches for model training: time-stepping prediction and treating time as an additional dimension, as illustrated in Fig. 5. Table 4 presents the results for Advection_time and NS2d_time, which correspond to using the time-stepping prediction method to solve the respective PDEs. On the other hand, Advection and NS3d represent treating time as an independent dimension to solve the corresponding equations. It is worth mentioning that the original DeepONet does not include a time-stepping prediction framework, hence no results are presented for this case. Our findings indicate that the time-stepping prediction approach struggles to achieve accurate predictions for certain PDEs, such as Advection. Conversely, treating time as a dimension and training the model accordingly proves to be more effective in solving such PDEs. This could be attributed to the utilization of the same Fourier layer for prediction at different time steps, which results in a decline in the prediction accuracy of the model for each step, particularly for discontinuous equations like Advection.

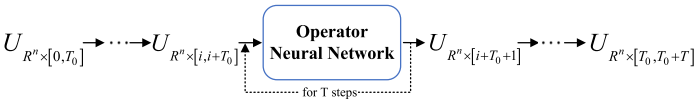
### 3.2. Physical information and data-driven

Whereas the neural operators exhibit significant potential in addressing partial differential equation problems, they are purely data-driven, necessitating a sufficient amount of training data. However, acquiring such data can be prohibitively expensive in practical applications, particularly when generating high-resolution and high-fidelity data. This scarcity of data presents a substantial challenge, impeding the widespread adoption of neural operators. Furthermore, purely data-driven neural operators face difficulties in generalizing to scenarios that differ from the conditions of the training data. As an alternative to data-driven methods, the PINNs [15] incorporate additional physical information constraints to achieve the solution of PDEs without data. When sufficient physical information is available, sufficiently accurate solutions to PDEs can be obtained without relying on extensive data. Hence, incorporating the physics-informed loss functions into neural operator frameworks is a promising approach to address the aforementioned issues.

#### 3.2.1. Physics-informed extensions of neural operators

In practical applications, problems can be categorized into three scenarios: the small dataset or data-scarce case where complete physical information is available, the case with partial data and physical information but missing key details, and the case of large data where no additional physical information is known. The second scenario is prevalent in practical PDE-solving applications, which has led to the development of hybrid approaches [76–78]. Hybrid neural operators seamlessly integrate data and physical information constraints, combining the advantages of neural operators to enhance solution speed while maintaining accuracy [14,79]. Wang et al. [76,80] were the pioneers in proposing a Physics-informed DeepONet (PI-DeepONet). Since the


(a) Time as a dimension




(b) Iterate step by step over time

**Fig. 5.** Time prediction algorithm architecture.

outputs of the DeepONet are differentiable for their input coordinates, physical constraints are incorporated into the DeepONet's loss function using automatic differentiation [81] to satisfy the constraints imposed by the underlying PDEs,

$$\mathcal{L}(\theta) := \mathcal{L}_{\text{data}}(\theta) + \mathcal{L}_{\text{physics}}(\theta), \tag{36}$$

where $\mathcal{L}_{\text{data}}(\theta)$ is the data supervisory constraint and $\mathcal{L}_{\text{physics}}(\theta)$ enforces the underlying PDEs constraints, which are denoted as follows:

$$\begin{aligned}\mathcal{L}_{\text{data}}(\theta) &= \frac{1}{NP}\sum_{i=1}^{N}\sum_{j=1}^{P}\left|G_\theta\left(\mathbf{u}^{(i)}\right)\left(\mathbf{x}_j^{(i)}\right) - G\left(\mathbf{u}^{(i)}\right)\left(\mathbf{x}_j^{(i)}\right)\right|^2,\\ \mathcal{L}_{\text{physics}}(\theta) &= \frac{1}{NQm}\sum_{i=1}^{N}\sum_{j=1}^{Q}\sum_{k=1}^{m}\left|\mathcal{N}\left(u^{(i)}\left(\mathbf{x}_k\right), G_\theta\left(\mathbf{u}^{(i)}\right)\left(\mathbf{x}_j^{(i)}\right)\right)\right|^2,\end{aligned} \tag{37}$$

where $\left\{\mathbf{u}^{(i)}\right\}_{i=1}^{N}$ denotes the N individual input functions sampled from U, $\mathbf{u}^{(i)}, \left\{\mathbf{x}_j^{(i)}\right\}_{j=1}^{P}$ is determined from the data observations, initial or boundary conditions, etc. $\left\{\mathbf{x}_j^{(i)}\right\}_{j=1}^{Q}$ is a set of points that can be randomly sampled in the $G(\mathbf{u}^{(i)})$ domain. Compared to the purely data-driven DeepONet, the PI-DeepONet achieves significant improvements in prediction accuracy, solution speed generalization performance, and data efficiency. And Wu et al. proposed a customized Physics-Informed Loss Interfaced Operator Network (IONet) [82] to solve parametric elliptic partial differential equations. IONet considers different coefficients, source terms, and boundary conditions as distinct input features, processes them separately, and uses a corresponding physics-informed loss function to constrain them.

Similarly, Li et al. [77] introduced the Physics-Informed Neural Operator (PINO) as an extension of FNO. PINO combines available data constraints and physical constraints to co-learn the solution operator of PDEs. It retains the discretization invariance of FNO and enables high-resolution predictions by integrating coarse-resolution training data with higher-resolution PDE constraints. Unlike PINN, which performs point-by-point optimization, PINO conducts optimization in function space, simplifying the optimization process. Additionally, PINO proposes a Fourier transform-based derivative method for solving PDEs in spectral space. Similar to the PI-DeepONet, PINO employs a loss function expressed as:

$$\mathcal{L}(\theta) := \mathcal{L}_{\text{data}}\left(G_\theta(\hat{a})\right) + \mathcal{L}_{\text{physics}}\left(\hat{a}, u^\dagger\right) + R(\hat{a}) \tag{38}$$

where $\mathcal{L}_{\text{pde}}$ is the physics-informed constraint, $\mathcal{L}_{\text{data}}$ is the data-supervised constraint, and $R(\hat{a})$ is the regularization term. The loss function in Eq. (38) represents the simplest case. It is worth noting that the choice of weights for $\mathcal{L}_{\text{physics}}$ and $\mathcal{L}_{\text{data}}$ also holds significance for model accuracy. PINO substantially enhances the generalization and physical validity of operator learning through the incorporation of known equation constraints. Nevertheless, since PINO employs the Fourier series representation, the exact gradient can only be applied to periodic functions to avoid the Gibbs phenomenon, which leads to suboptimal optimization. Consequently, a novel extension is proposed to enable PINO to be employed with non-periodic functions while maintaining numerical accuracy [83]. It is worth noting that the most common approach to compute derivatives during PINN optimization is to utilize the autograd capability of neural networks. Thus, the introduction of new derivative methods in neural operator architectures [77] currently is a primary avenue for enhancing the integration of neural operators with PINNs. Similar to PINO, N. et al. [78] proposed Physics-informed WNO (PI-WNO), which incorporates stochastic projection-based gradients to enhance the accuracy of derivative calculations (see Fig. 6).

### 3.2.2. Extensions and applications of hybrid neural operator architectures

Neural operators and their extensions have demonstrated excellent performance in numerous applications. However, they often necessitate a substantial volume of data to attain the desired accuracy. Correspondingly, PINNs compensate for the challenges associated with acquiring large amounts of data and effectively enhance both accuracy and generalization, but they are often difficult to train. The combination of physical information and data-driven methods can effectively compensate for their respective limitations. Incorporating physical information into training results in highly accurate operators that can be generalized across different data resolutions. Hybrid neural operators are frequently employed in the context of multi-fidelity learning [84–86]. Training high-dimensional operators necessitates a significant quantity of costly high-fidelity data. In practical applications, employing numerical methods to generate high-fidelity solutions can result in prohibitively high expenses. In such scenarios, it can be advantageous to complement high-fidelity datasets with low-fidelity counterparts and integrate corresponding additional physical information, which is known as multi-fidelity learning [87]. Physical-informed multi-fidelity learning leverages physical information as a means of correcting low-fidelity data, which is calibrated using high-fidelity physical-informed constraint to produce accurate solutions with high-fidelity.

Similar to the concept of transfer learning [88,89], DeepM&Mnet fine-tunes the pre-trained operator model using physical information and a small amount of data. In new scenarios with only a limited number of data points, it can still achieve satisfactory accuracy by retraining additional networks to fine-tune the output of neural operators [33, 90]. Furthermore, Mao et al. [90] proposed utilizing the pre-trained DeepONet series to enhance the accuracy and generalization of model predictions in new scenarios. Despite the availability of only partial variable data, this method can still generate prediction results that meet the accuracy requirements under the constraints of prior information.

Moreover, the combination of physical information and data-driven methods is also utilized to address extrapolation problems. Neural networks are typically limited to solving interpolation problems, meaning

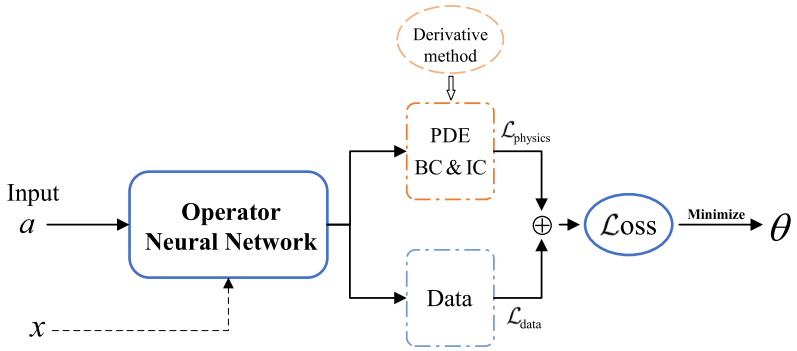



**Fig. 6.** Physical information and data-driven hybrid neural operator network framework.

that their predictions are accurate only for inputs within the range of the training set. If a new input falls outside this domain (i.e., extrapolation), the prediction is likely to be inaccurate or fail [91]. To obtain predictions beyond the function space, the physical-informed neural operator [92,93] can also be fine-tuned by incorporating PDE losses into the pre-trained DeepONet. This approach effectively enhances the performance of the operator method in handling extrapolation.

### 3.2.3. Experiments

In this section, we select four datasets with varying dimensional and spatial characteristics to compare and analyze the performance of FNO, WNO, and DeepONet under three different training scenarios: (1) purely physical-informed training (w/o data); (2) purely data-driven training (only data); and (3) a hybrid approach combining physical information and data (with data). We assess the performance of the framework by measuring the $L_2$ relative errors predicted by various neural operators on dataset instances with distinct features. Furthermore, we examine the robustness of neural operators when subjected to noise interference, comparing their performance in three scenarios. Detailed information about the experimental datasets can be found in Appendix A.2. The parameter settings can be seen in Appendix B.

To examine the robustness of the pure data, pure physical information, and hybrid neural operators to noise, we conducted tests in the following scenarios on Burgers' dataset: (1) training on noisy data and testing with noise-free and noisy data, and (2) training on noisy data and testing with noise-free and noisy data. For the creation of noisy data $a'$ from input function $a$, we employed the following settings:

$$a(x)' = a(x) + 0.1 \cdot \|a\|_{\infty}\xi, \tag{39}$$

where $\xi \sim \mathcal{N}(0,1)$ follows an independent normal distribution. Then, to evaluate the impact of PDE constraints at different resolutions on the across-resolution performance of representative neural operators, we performed tests that were trained using low-resolution data and different-resolution PDE losses. However, due to DeepONet's lack of discretization invariance and the adaptive truncation of input data during wavelet packet invocation in WNO, which prevents simultaneous adaptation to different resolution inputs, we only conducted experiments using the PINO.

Finally, we discuss the interpolation (In.) and extrapolation (Ex.+) performance of the two methods by testing PDE datasets sampled from different GRF spaces (different correlation lengths $l$).To test the Ex.+, the Burgers' equation correlation length of train datasets is $l_{train} = 1.0$, and the test dataset of 100 functions with $l_{test} = 0.6$. For the advection equation, the $u(x)$ is randomly sampled from a GRF with an RBF kernel with the correlation length $l_{train} = 0.5$, and the test dataset of 100 functions with $l_{test} = 0.2$. And Diffusion-reaction dynamics' term $u(x)$ is randomly sampled from a GRF with an RBF kernel with the correlation length $l_{train} = 0.5$, and the test dataset of 100 functions with $l_{test} = 0.2$.

*Discussions:* The prediction error results for FNO, DeepONet, and WNO under three different training scenarios are presented in Table 5. We do not show the full results of PI-DeepONet on the Allen-Cahn equation because it fails to achieve good accuracy. Previous research has indicated that minimizing the loss of PINNs can be challenging, potentially resulting in incorrect learning of the solutions [94,95]. As shown in Table 5, PI-WNO encounters difficulty in solving the Burgers' equation, indicating that training neural operators solely based on the control equation in the loss function may result in inaccurate predictions in practical applications. Despite the challenges faced by PI-WNO in solving the Burgers' equation, the inclusion of additional physical information can enhance the accuracy of WNO in data-driven training scenarios. The experimental results reveal that the trained hybrid models can achieve comparable or even superior accuracy when compared to pure data-driven or pure physics-informed methods in the Burgers', Diffusion-reaction, and Allen-Cahn equations. However, the loss function of hybrid neural operators typically comprises multiple terms, making it a highly non-convex optimization problem. The terms within the loss function may compete with each other during training [14]. Consequently, the incorporation of physical information may interfere with the learning process of the neural operators for the advection equation, resulting in a decrease in accuracy. Additionally, in this experiment, finite difference methods were employed to solve the PDE constraints, aiming to mitigate the interference arising from the use of different derivative methods. Achieving improved accuracy is possible by employing more suitable derivative methods in practical applications.

The training and testing results of pure data, pure physical information, and hybrid neural operators on both noisy datasets and clean datasets are presented in Table 6. Except for WNO, which cannot be solved under the pure physics-informed constraint, all operators exhibit solution errors of less than 10%.

To evaluate the effect of the hybrid approach on across-resolution applications, we apply the PINO to datasets of the Burgers', Diffusion-reaction, and Allen-Cahn equations individually. We train the model using data constraint at low spatial resolutions ($32 \times 25$, $25 \times 25$, $33 \times 33$) and varying resolutions of PDE losses (the low, medium and high PDE constraint resolutions: $\times 1$, $\times 2$ and $\times 4$). The results are presented in Table 7. PINO achieves a test accuracy of 0.08% when trained on the $32 \times 25$ resolution dataset. However, when tested on across-resolution scenarios of $64 \times 50$ and $128 \times 100$, the accuracy drops to 3.13% and 3.47%, respectively. When the $128 \times 100$ PDE loss is included in the across-resolution testing, the operator's accuracy improves significantly to 0.52% and 0.69%, respectively. We observe that incorporating low-resolution PDE constraints does not lead to a significant improvement in the across-resolution accuracy of the neural operator. However, including high-resolution PDE losses in training enhances the operator's generalization capabilities for across-resolution scenarios. Therefore, when applying neural operators to practical across-resolution tasks, it is recommended to utilize high-resolution PDE constraints to enhance both model solution accuracy and generalization. Furthermore, we observed that purely physics-informed neural operators often exhibit

**Table 5**
Prediction error results of PINO, PI-WNO, and PI-DeepONet. The prediction error corresponds to the $L_2$ relative error of the optimized model after training.

| | | Burger's (%) | Advection (%) | Diffusion_reaction (%) | Allen-Cahn (%) |
|---|---|---|---|---|---|
| PINO | w/o data | 0.60±0.03 | 3.20±0.00 | 0.3±0.00 | 1.30±0.30 |
| | **with data** | **0.15±0.04** | **0.70±0.02** | **0.09±0.00** | **0.23±0.01** |
| | only data | 0.20±0.06 | 0.70±0.03 | 0.10±0.00 | 0.20±0.01 |
| PI-DeepONet | w/o data | 1.20±0.08 | 4.30±0.05 | 0.70±0.00 | – |
| | **with data** | **0.28±0.00** | 1.46±0.02 | **0.54±0.16** | – |
| | only data | 0.30±0.01 | **1.40±0.01** | **1.10±0.10** | 41.50±0.00 |
| PI-WNO | w/o data | 73.30±15.32 | 8.90±0.03 | 2.10±0.14 | 2.5±0.05 |
| | **with data** | **3.01±0.09** | 2.35±0.02 | 0.88±0.15 | **1.09±0.00** |
| | only data | 4.80±0.11 | **1.50±0.01** | **0.80±0.29** | 1.20±0.00 |

**Table 6**
Prediction error results of PINO, PI-WNO, and PI-DeepONet under 0.1 noise. The prediction error corresponds to the $L_2$ relative error of the optimized model after training.

| | | | Train error (%) | Test (clean) (%) | Test (noisy) (%) |
|---|---|---|---|---|---|
| Burger training clean | PINO | w/o data | 0.70±0.04 | 0.67±0.05 | 2.85±0.10 |
| | | with data | 0.08±0.00 | 0.08±0.00 | 2.84±0.00 |
| | | only data | 0.08±0.00 | 0.08±0.00 | 2.87±0.00 |
| | PI-DeepONet | w/o data | 3.46±0.22 | 1.47±0.29 | 1.92±0.02 |
| | | with data | 0.64±0.01 | 0.28±0.00 | 4.85±0.54 |
| | | only data | 0.76±0.03 | 0.31±0.00 | 4.99±0.78 |
| | PI-WNO | w/o data | 86.35±28.57 | 86.27±28.9 | 86.39±28.8 |
| | | with data | 2.07±0.03 | 2.16±0.03 | 7.44±0.14 |
| | | only data | 2.34±0.03 | 2.43±0.03 | 7.53±0.14 |
| Burger training noisy | PINO | w/o data | 3.25±0.00 | 0.69±0.01 | 3.15±0.00 |
| | | with data | 2.37±0.00 | 0.76±0.00 | 2.55±0.00 |
| | | only data | 2.37±0.00 | 0.74±0.00 | 2.55±0.00 |
| | PI-DeepONet | w/o data | 5.5±0.09 | 1.59±0.19 | 3.56±1.71 |
| | | with data | 4.65±0.26 | 1.59±0.00 | 2.3±0.42 |
| | | only data | 4.49±0.07 | 1.14±0.00 | 3.44±0.71 |
| | PI-WNO | w/o data | 91.19±8.48 | 91.14±9.26 | 86.79±9.68 |
| | | with data | 4.53±0.07 | 3.00±0.05 | 4.87±0.12 |
| | | only data | 4.68±0.07 | 3.23±0.07 | 4.88±0.04 |

**Table 7**
Use fixed low-resolution data and different resolution PDE losses to perform across-resolution testing after training on the datasets.

| | PDE resolution | Test resolution | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Burgers' (%) | | | Diffusion_reaction (%) | | | Allen-Cahn (%) | |
| | | 32*25 | 64*50 | 128*100 | 25*25 | 50*50 | 100*100 | 33*33 | 65*65 |
| w/o data | High | **0.78±0.10** | **0.58±0.11** | **0.57±0.10** | 29.39±4.22 | 14.68±1.74 | **0.13±0.00** | – | – |
| | Medium | 1.36±0.14 | 1.31±0.13 | 1.34±0.12 | 19.03±0.83 | **3.56±0.01** | 18.92±1.24 | **1.06±0.04** | **0.78±0.03** |
| | Low | 16.74±1.18 | 12.34±0.51 | 13.01±0.59 | **11.36±0.16** | 24.91±0.84 | 33.24±0.90 | 2.26±0.06 | 2.11±0.08 |
| with data | High | **0.08±0.03** | **0.52±0.05** | **0.69±0.05** | **0.05±0.00** | 5.83±0.24 | **1.2±0.01** | – | – |
| | Medium | 0.12±0.02 | 0.91±0.03 | 2.51±0.05 | 0.05±0.00 | **3.43±0.04** | 19.95±0.92 | **0.37±0.03** | **0.28±0.01** |
| | Low | 0.09±0.01 | 3.13±0.08 | 3.67±0.11 | 0.05±0.00 | 37.09±1.2 | 42.91±1.31 | 0.38±0.01 | 0.31±0.00 |
| only data | – | 0.08±0.02 | 3.13±0.08 | 3.67±0.08 | 0.05±0.00 | 40.69±0.54 | 45.13±0.39 | 0.39±0.01 | 0.32±0.01 |

superior performance on high-resolution output tests, regardless of the resolution of PDE constraints when training. This observation presents an intriguing and worthwhile topic for further discussion.

Finally, because The input functions for training and testing are generated from Gaussian random fields (GRFs), if the correlation length for training ($l_{\text{train}}$) is different from the correlation length for testing ($l_{\text{test}}$), it is extrapolation (Ex.+). We compare the extrapolation performance of physical-informed and data-driven FNO, DeepONet, and WNO by testing PDE datasets sampled from different GRF spaces. The results can be seen in Table 8. Judging from the L2 relative error alone, it seems that FNO has achieved the best extrapolation effect. However, by comparing the difference between interpolation and extrapolation prediction errors of data-driven and physical information, the PDE constraint directly reveals the underlying information of the PDE, it has better extrapolation ability. Therefore, using PDEs constraint to fine-tune the pre-trained model for application in out-of-distribution environments is widely used.

### 3.3. Complex systems

PDEs are extensively employed for characterizing complex systems across diverse fields such as physics, chemistry, and biology. Complex systems frequently encompass multiple spatial and temporal dimensions, exhibiting characteristics like nonlinearity, coupling, and inhomogeneity [1,33,90,96]. Solving complex systems using traditional finite element methods incurs significant computational expenses. Pre-training neural operators on existing datasets enable their extension to other input coordinates, resulting in a substantial improvement in the speed of solving complex PDEs. Despite the promising results demonstrated by neural operators in PDE solving, they encounter several challenges in practical applications, particularly within complex systems. These challenges primarily encompass irregular grids, multiple inputs, and multiscale problems. To address these challenges, researchers have proposed numerous enhancements in neural operators. Additionally,

**Table 8**
L2 relative error of different methods(data-driven and physical information) for three benchmarks in interpolation and extrapolation.

| | Burger's (%) | | Advection (%) | | Diffusion_reaction (%) | |
|---|---|---|---|---|---|---|
| | In. | Ex.+ | In. | Ex.+ | In. | Ex.+ |
| FNO | **0.38**$_{\pm 0.02}$ | **1.10**$_{\pm 0.02}$ | **0.18**$_{\pm 0.00}$ | **2.51**$_{\pm 0.06}$ | **0.05**$_{\pm 0.00}$ | **1.00**$_{\pm 0.04}$ |
| PINO | 2.7$_{\pm 0.95}$ | 2.98$_{\pm 0.85}$ | 1.77$_{\pm 0.11}$ | 4.21$_{\pm 0.18}$ | 0.20$_{\pm 0.03}$ | 1.99$_{\pm 0.16}$ |
| DeepONet | 1.72$_{\pm 0.17}$ | 5.4$_{\pm 0.05}$ | 0.99$_{\pm 0.02}$ | 6.31$_{\pm 0.28}$ | 0.40$_{\pm 0.01}$ | 9.60$_{\pm 0.22}$ |
| PI-DeepONet | 4.68$_{\pm 0.10}$ | 12.69$_{\pm 0.57}$ | 3.10$_{\pm 0.14}$ | 8.13$_{\pm 0.47}$ | 4.91$_{\pm 2.00}$ | 25.82$_{\pm 1.90}$ |
| WNO | 2.94$_{\pm 0.01}$ | 8.15$_{\pm 0.01}$ | 0.40$_{\pm 0.00}$ | 5.22$_{\pm 0.07}$ | 0.20$_{\pm 0.03}$ | 1.97$_{\pm 0.26}$ |
| PIWNO | 89.29$_{\pm 1.99}$ | 87.41$_{\pm 2.88}$ | 3.36$_{\pm 0.01}$ | 9.31$_{\pm 0.01}$ | 1.19$_{\pm 0.16}$ | 9.01$_{\pm 0.38}$ |

uncertainty quantification is a key challenge, and recent developments in Bayesian and probabilistic neural operators have shown great potential in quantifying the uncertainty in predictions, enhancing their reliability. Furthermore, real-world applications across three major domains – climate science, engineering, and biomedicine – are driving further advancements and applications of neural operators. Researchers have proposed numerous enhancements to address these challenges, aiming to improve the robustness and scalability of neural operators in real-world scenarios.

### 3.3.1. Irregular grids

In practical applications involving complex problems with irregular geometric grids, the solutions to partial differential equations frequently exhibit a significant degree of variability within the problem domain. However, a significant challenge arises in addressing the irregular grid problem as many high-performance neural operators are limited by the requirement for inputs and outputs to possess the same discretization on regular equidistant grid points [19,21,31,37].

Due to its implementation using Fast Fourier Transform (FFT), FNO is limited to application within rectangular domains featuring uniform grids. Initially, to apply FNO to irregularly shaped regions, the problem was solved by interpolating the irregular domains into finer regular grid points [97]. Nevertheless, this approach can result in substantial interpolation errors, particularly when dealing with nonlinear partial differential equations. Subsequently, Lu et al. [64] proposed a domain extension technique that involves embedding the region to be solved within a larger rectangular region to overcome this limitation. Nevertheless, this approach proves highly inefficient and computationally wasteful when dealing with highly irregular grids. To address this, Li et al. [98] proposed Geo-FNO, which applies FNO extensions to irregular grids by learning the mapping from irregular to regular grids. Additionally, Chen et al. [49] introduced the Laplace Neural Operator for complex geometries (Geo-LNO), which exhibits the ability to be applied to complex geometries while preserving discretization invariance.

While FNO is limited to predicting solutions within grids that possess the same input function, DeepONet can predict solutions at any location. However, DeepONet lacks the advantage of discretization invariance. Consequently, a novel mesh-free neural operator called Basis Enhanced Learning Network (BelNet) [18] is proposed. BelNet is built upon the universal approximation theorem [6] and the subsequent expression:

$$G(a)(x) \approx G_\theta(a)(x) = \sum_{k=1}^{K} \sigma_x \left( \left(q^k\right)^\top x + b_{\underline{x}}^k \right) \times \sigma_a \left( \hat{a}^\top W_{x_a}^{2,k} \left( \sigma_{x_a} \left( W_{x_a}^{1,k} x_a + b_{x_a}^k \right) \right) \right), \tag{40}$$

where $x_a \in K_2 \subset \mathbb{R}^d$, $a \in V$, and $\hat{a} = \left[a\left(x_{a,1}\right), \ldots, a\left(x_{a,N}\right)\right]^\top \in \mathbb{R}^N$. BelNet integrates the advantageous features of DeepONet and FNO, enabling it to handle irregular mesh inputs and outputs at arbitrary locations while simultaneously maintaining discretization invariance.

Transformer offers a versatile approach to effectively capturing and encoding spatial information. Despite the proposal of GK-Transformer, the input and output must be within the same domain. To address this limitation, a novel attention-based neural operator framework called Operator Transformer (OFormer) [99] is proposed. OFormer comprises self-attention, cross-attention, and a set of MLPs, enabling the construction of a versatile operator learning architecture to arbitrarily query point locations and accommodate varying numbers of input points. Furthermore, it exhibits flexibility in adapting to different types of grids.

### 3.3.2. Multi-scale

Numerical simulations in various fields, including chemistry, astronomy, physics, and climate, are frequently characterized by their inherent complexity. These systems typically consist of multiple levels or scales, resulting in the division of the entire system field into distinct sub-domains with varying physical characteristics [100]. For instance, at the molecular level, chemical reactions occur among atoms and molecules. In contrast, at the macro level, these molecules form substances, and interactions and macroscopic movements occur between them. Effectively capturing and modeling the coupling and interactions between different scales pose significant challenges in addressing multi-scale PDE problems. Simulating multi-scale problems within the system also entails computationally expensive calculations, thereby intensifying the complexity of learning multi-scale functions.

Building upon the impressive performance of neural operators in PDE solving, the multi-scale concept is incorporated into DeepONet, leading to the development of Multi-scale DeepONet [101]. As evident from the preceding statement, DeepONet does not explicitly define the architecture of its branch and trunk networks. Consequently, in multi-scale DeepONet, a multi-scale deep neural network is employed in both the branch and trunk networks, resulting in numerous distinct combinations. For instance, if the branch network is configured as a multi-scale deep neural network and the trunk network is a feedforward neural network, the configuration can be expressed as follows:

$$u(x) = G(a)(x) \approx \sum_{k=1}^{N} \sum_{i=1}^{M} c_i^k \sigma_b \left( \sum_{j=1}^{m} \xi_{ij}^k a\left(x_j\right) + \theta_i^k \right) \cdot \sigma_t \left( \omega_k \cdot x + \zeta_k \right). \tag{41}$$

The resulting multi-scale DeepONet can effectively represent multi-scale operators in oscillatory responses. Furthermore, Yin et al. [102] introduced a concurrent coupling method to integrate neural operators with traditional numerical solvers for multi-scale modeling. Due to the significant computational costs related to traditional multi-scale modeling, DeepONet is employed as a substitute for high-fidelity models in combination with traditional numerical solvers. This framework substantially reduces the computational overhead associated with multi-scale simulations in multi-scale modeling.

Building upon the remarkable performance of FNO in single-phase flow problems, U-FNO [103] is introduced as an extension of the FNO-based architecture to address multiphase flow problems. U-FNO incorporates a two-step U-Net in the Fourier layer, enabling more effective capture of interactions and phase transition phenomena at different scales in multiphase flows, leading to higher accuracy and speed. However, the inclusion of U-Net in the network causes U-FNO to lose discretization invariance. Implicit Fourier Neural Operators (IFNOs) [104] have been employed in multiscale problems, representing the solution operator as an implicitly defined mapping. IFNOs

offer substantial improvements in accuracy and stability for solving multiscale problems compared to FNO. While the multi-scale DeepONet and FNO have been proposed, purely data-driven operators are computationally expensive for large-scale multiscale dynamics simulations. Therefore, Lütjens et al. [105] propose a new Multiscale Neural Operator (MNO). The MNO integrates the neural operator with a large-scale multiscale model. In this framework, the large-scale multiscale model rapidly simulates the propagation state, leveraging the neural operator module to learn the cumulative error during propagation. The large-scale multiscale networks can incorporate physical information [15] through a loss function to improve their performance. Additionally, due to the absence of discretization invariance in DeepONet, the MNO prefers to select FNO, and it leverages the known large-scale physical information to achieve a mesh-independent differentiable solver.

In the context of multiscale problems, many existing neural operators primarily focus on capturing the smooth part of the solution space, which poses challenges in effectively capturing its intrinsic multiscale features. Liu et al. [106] propose the Hierarchical Transformer Model (HT-Net), which is a neural operator based on the hierarchical transformer. It is designed to learn the solution mapping for multiscale PDEs effectively. HT-Net utilizes a hierarchical discretization strategy and treats spatial interactions by considering geometric scales and locality separately. This approach improves the capability to capture oscillatory features in the multiscale solution space, leading to enhanced accuracy and generalization of models. Similarly, The Hierarchical Attention Neural Operator (HANO) [107] automatically decomposes the input–output mapping into a hierarchical structure and incorporates nested feature updates through hierarchical local aggregation of self-attention. This approach ensures controllable linear computational cost and enhances the capability to capture oscillatory features in a multi-scale solution space.

### 3.3.3. Multiple inputs

Neural operators have been successfully applied to a diverse array of real-world problems, demonstrating commendable performance [108–111]. However, numerous complex problems require modeling input functions comprising multiple data types, such as boundary shapes, global parameter vectors, etc. Nevertheless, current neural operators such as DeepONet, FNO, etc., are all designed to learn only mapping defined on a single Banach space(i.e., the input to the operator is a single function). Consequently, designing novel neural operators with the flexibility to handle diverse input types is an ongoing challenge.

Addressing the limitations of neural operators when confronted with multi-input problems, Jin et al. [32] introduced Multi-input DeepONet (MIONet). The architecture of MIONet closely resembles that of DeepONet. However, it comprises $n$ independent branch networks and a trunk network. The $i$th branch network, denoted as $\tilde{\mathbf{g}}_i$, encodes the corresponding input function $v_i$, while the trunk network, represented by $\tilde{\mathbf{f}}$, encodes the input function $x$. MIONet manifests in two slightly distinct versions, which are as follows:

$$\begin{aligned}
\text{high-rank:}\tilde{G}\left(a_1,\ldots,a_n\right)(x) &= \underbrace{\tilde{\mathbf{f}}(x)}_{\text{trunk}}\langle\underbrace{\tilde{\mathbf{g}}_1\left(\varphi_{q_1}^1\left(a_1\right)\right)}_{\text{branch}_1},\ldots,\\
&\quad\underbrace{\tilde{\mathbf{g}}_n\left(\varphi_{q_n}^n\left(a_n\right)\right)}_{\text{branch}_n}\rangle+b,\\
\text{low-rank:}\tilde{G}\left(a_1,\ldots,a_n\right)(x) &= S(\underbrace{\tilde{\mathbf{g}}_1\left(\varphi_{q_1}^1\left(a_1\right)\right)}_{\text{branch}_1}\odot\cdots\odot\underbrace{\tilde{\mathbf{g}}_n\left(\varphi_{q_n}^n\left(a_n\right)\right)}_{\text{branch}_n}\\
&\quad\odot\underbrace{\tilde{\mathbf{f}}(x)}_{\text{trunk}})+b,
\end{aligned}\tag{42}$$

where $\odot$ represents the Hadamard product, and $S$ denotes the sum of all components within the vector. Furthermore, they established the approximation theory for successive multi-input DeepONet [32]. While U-FNO has attained state-of-the-art performance in highly inhomogeneous geological formations, its computations are prohibitively expensive and slow. Therefore, a Fourier-enhanced multi-input neural operator, known as Fourier-MIONet [112], was developed based on MIONet. Fourier-MIONet specializes in learning the solution operator for multiphase flow problems in porous media. In comparison to U-FNO, Fourier-MIONet conserves 85% of CPU memory and 64% of GPU memory while achieving a training speed boost of 3.5 times. This substantial reduction in computational cost does not compromise prediction accuracy. Similar to the idea of MIONet, Zhang et al. proposed the distributed deep neural operator (D2NO) [113], which uses different branch networks to sample different input functions. It improves the efficiency of neural operator solving while maintaining the accuracy of prediction.

When employing DeepONet to solve stochastic differential equations (SDEs), since the branch network input is a set of N characteristic functions, the dimension of the trunk network input is the sum of the dimensions of the physical space and the random space. Therefore, as the dimensionality of the SDEs problem increases, it will become infeasible to use the original DeepONet to solve it. This renders the utilization of vanilla DeepONet infeasible for problem-solving. Consequently, Zhang et al. [114] devised a multi-resolution auto-encoder DeepONet (MultiAuto-DeepONet), to address this issue by facilitating nonlinear dimensionality reduction and stochastic operator learning. MultiAuto-DeepONet tackles high-dimensional SDEs and possesses the capability to process inputs from diverse domains. This capability significantly enhances problem-solving efficiency.

*General architecture:* While previous works have addressed individual challenges through different architectures, no attempt has been made to construct network architectures that can simultaneously handle these challenges. For instance, MIONet can manage multiple input functions but proves inadequate for multi-scale problems. Consequently, Hao et al. [31] endeavored to develop a general neural operator transformer (GNOT) that can simultaneously address the three challenges of irregular grids, multiple inputs, and multi-scale. GNOT employs distinct MLPs to encode input query points and input functions. It subsequently updates the features of query points through a heterogeneous normalized cross-attention layer and a normalized self-attention layer. A gate network utilizes the geometric coordinates of query points to calculate a weighted average of multiple expert feed-forward neural networks. Finally, GNOT outputs the processed features after subjecting them to N layers of the attention block. GNOT outperforms various other models across multiple datasets, yielding superior results. However, it is essential to note that the inclusion of the attention mechanism increases the model's memory and computation requirements. Hence, GNOT proves particularly advantageous for complex systems or large datasets.

### 3.3.4. Uncertainty quantification

In practical applications, the process of operator learning is inevitably affected by various sources of uncertainty. These include inherent noise in the data – such as measurement errors, limitations in sensor precision, or missing observations – as well as uncertainty arising from the model itself. Often, models are constructed under simplifying assumptions that may not fully capture the complexity of the underlying physical or dynamical systems. Moreover, in tasks involving long-term prediction, such as in chaotic dynamical systems, the reliability of forecasts can rapidly deteriorate. In light of these challenges, a rigorous understanding and quantification of predictive uncertainty is essential for building robust and trustworthy neural operator models. As neural operators – particularly DeepONets and Fourier Neural Operators (FNOs) – gain prominence for modeling complex parametric PDEs and dynamical systems, the need for reliable and efficient uncertainty quantification (UQ) has become increasingly critical. This is especially

true in high-stakes or data-scarce applications, where models must not only be accurate but also trustworthy.

*Bayesian Formulations and Posterior Approximation.* Several works have extended deterministic neural operators with Bayesian inference to quantify epistemic uncertainty. The Bayesian DeepONet (B-DeepONet) with replica exchange Langevin diffusion (reLD) [115] introduces a two-particle training approach that jointly explores and exploits the loss landscape, improving robustness to noisy data. Similarly, the Variational Bayes DeepONet (VB-DeepONet) [116] employs variational inference to approximate complex posterior distributions, improving generalization while offering predictive uncertainty. In a more recent development, the Ensemble Kalman Inversion (EKI) method is applied to DeepONets for derivative-free, noise-resilient UQ [117]. This approach allows efficient ensemble training and uncertainty estimation, even under limited and noisy data, and offers strong parallelizability and adaptability to large datasets. In addition to these training-time methods, lightweight post-hoc strategies have also been explored. The LUNO framework [118], for instance, adopts a Laplace approximation in the neural weight space and propagates uncertainty through a linearized version of the pretrained operator model. This yields a function-valued Gaussian process approximation with negligible computational overhead, making it especially suitable for large-scale deployments or pretrained systems where retraining is impractical.

*Gaussian Process-Based Hybrid Methods.* Blending neural operators with Gaussian Processes (GPs) offers a principled and interpretable route for uncertainty quantification. The Neural Operator-induced Gaussian Process (NOGaP) [119] framework embeds operator learning within the GP structure, resulting in quantifiable and well-calibrated uncertainty estimates. Another hybrid method [120] defines a neural operator-embedded kernel for GPs, trained jointly with GP hyperparameters using a stochastic dual descent algorithm. This enables scalable, resolution-independent inference across high-dimensional parametric PDE problems. Building on this line, a recent framework [121] leverages GPs to approximate operator bilinear forms, supporting both zero-mean and neural-mean settings where DeepONet or FNO serve as the mean function. The model supports both data-driven and physics-informed training via maximum likelihood and PDE residuals, and enables zero-shot prediction through proper kernel initialization. Structured kernels and Kronecker representations further enhance efficiency and scalability across multi-output tasks.

*Generative and Probabilistic Modeling Approaches.* Beyond posterior approximation, recent methods frame operator learning as a generative modeling task. The Probabilistic Neural Operator (PNO) [122] introduces uncertainty directly into training via proper scoring rules, allowing for the generation of calibrated function-space distributions. This enables robust prediction of chaotic systems and long-term dynamics. Likewise, diffusion-based generative models have been explored as neural operators capable of capturing multimodal predictive distributions and addressing ill-posed inverse problems [123]. These models learn both forward and inverse mappings and produce uncertainty estimates through sampling diversity.

In summary, uncertainty quantification in neural operators has evolved along several complementary fronts: Bayesian training methods enhance model robustness and posterior sampling; GP-integrated architectures combine expressivity with principled uncertainty; and generative frameworks provide deep probabilistic modeling capabilities. These diverse innovations collectively enable neural operators to become not only powerful but also trustworthy surrogates for real-world scientific and engineering tasks.

### 3.3.5. Real world applications

Neural operators have matured into powerful, general-purpose surrogates for complex physical systems, delivering dramatic speed-ups and robust generalization across climate science, engineering, and biomedical domains. Below, we synthesize key advances organized by application area and thematic innovation.

**Climate Applications of Operator Learning.** Operator learning has revolutionized climate and environmental sciences, providing more scalable, accurate, and efficient alternatives to traditional physics-based models. In particular, neural operators, such as Fourier Neural Operators (FNOs), have enabled breakthroughs in global weather forecasting, carbon capture and storage (CCS), climate downscaling, air quality forecasting, and environmental modeling.

*Weather Forecasting.* Wavelet Neural Operator (WNO) [17] combines wavelet transforms with integral kernels, offering improved spatial and frequency resolution. This method is particularly effective for tracking complex dynamics, such as Earth's air temperature, by providing high resolution in both time and space. A major line of work focuses on global weather forecasting, where the spherical geometry of the Earth imposes unique challenges. Traditional Fourier Neural Operators, while powerful for capturing long-range dependencies, suffer from artifacts when applied to spherical domains. To address this, Spherical Fourier Neural Operator (SFNO) [124], and further improvements such as Spherical Neural Operator (SNO) [125] and Spherical Multigrid Neural Operator (SMgNO) [126] have been developed. These methods introduce spherical convolutions and multigrid frameworks to accurately model weather dynamics while significantly reducing computational costs. Notably, SMgNO achieves superior performance with only a fraction of the computational resources compared to previous spherical models. Additionally, DeepONet applied to latent spaces [19] enables real-time predictions of nonlinear, multiscale systems such as atmospheric flows. By operating in low-dimensional latent spaces, this approach significantly reduces computational costs while maintaining high prediction accuracy. LOCA (Learning Operators with Coupled Attention) [127] improves global weather forecasting by mapping sparse air temperature data to surface pressure fields. Its coupled attention mechanism boosts prediction accuracy in data-scarce scenarios, offering a valuable tool for climate modeling. Further improving global forecasts, FourCastNet [128], leveraging adaptive FNOs, enables medium-range, high-resolution global weather predictions five orders of magnitude faster than traditional numerical weather prediction (NWP) models, making ensemble forecasting of extreme weather events feasible at unprecedented scales. In addition, ModAFNO [129] introduces an interpolation model that reconstructs atmospheric states at higher temporal resolutions, demonstrating its capability to produce intermediate time steps with remarkable accuracy. This model improves the reconstruction of extreme weather events, thereby addressing limitations posed by coarse temporal resolution in conventional models.

*Downscaling & Super-Resolution.* Efforts have also been made in climate downscaling, crucial for generating fine-grained climate data from coarse simulations. Operator learning models such as the Spatiotemporal Multimodal Fusion Operator with a State-Query Coupled Kernel (SMCK) [130] address challenges in multimodal and spatiotemporal fusion for regional climate analysis. Meanwhile, FNO-based downscaling frameworks [131,132] demonstrate "zero-shot" capabilities – training on low-resolution data and accurately predicting arbitrary high-resolution outputs without retraining – offering a promising alternative to conventional high-cost dynamical downscaling approaches.

*Bias Correction & Air Quality.* To further bridge the gap between simulation and data-driven correction, operator learning methods have been introduced for bias correction in climate simulations. DeepONet-based surrogate models [133] have successfully replaced traditional nudging modules in Earth system models, learning high-dimensional correction tendencies efficiently through latent representations. In terms of air quality forecasting, which is tightly linked to climate and urban planning, the Complex Neural Operator for Air Quality (CoNOAir) [134] offers real-time, high-fidelity predictions of CO concentrations across major cities, outperforming FNOs especially in capturing extreme pollution events.

*Carbon Storage.* Beyond atmospheric modeling, operator learning has been increasingly applied to geological carbon storage (GCS) and carbon capture and storage (CCS) applications. Predictive models such

as the Nested Fourier Neural Operator (Nested FNO) [135] and the Improved Neural Operator (INO) [136] facilitate real-time, high-resolution 3D $CO_2$ storage simulations, addressing computational bottlenecks and supporting the safe deployment of CCS technologies on a global scale. Further improving efficiency, multi-fidelity FNO models [137] allow for the use of coarser datasets to train accurate models, reducing data generation costs by 81% while maintaining high accuracy in pressure field predictions. Gabor-Filtered Fourier Neural Operator (GFNO) [138] improves the expressiveness of FNOs, enhancing their accuracy for large-scale CCS and other environmental simulations by reducing error and computational cost. A Fourier Neural Operator-based surrogate model [139] has been developed to simulate $CO_2$ plume migration in realistic 3D subsurface environments, achieving $O(10^5)$ computational speedup with minimal accuracy loss.

*Multi-Scale and Multi-Physics Environmental Modeling.* Lastly, efforts to model coastal hydrodynamics [140] and upper atmospheric temperature profiles [141] with neural operator networks demonstrate the broad versatility of these methods in tackling various multi-scale, multi-physics problems within the climate and environmental sciences. Moreover, Neural Manifold Operator (NMO) [142], which learn the intrinsic dimensions of physical systems, has demonstrated superior accuracy and physical consistency when applied to climate and ocean systems, providing a robust framework for future environmental modeling.

Collectively, these innovations not only showcase the effectiveness of operator learning in enhancing prediction accuracy, computational efficiency, and scalability in climate-related tasks but also underscore its potential in supporting critical infrastructure planning, environmental monitoring, and climate change mitigation efforts.

**Engineering Applications of Neural Operators.** Neural operators, such as DeepONet and Fourier Neural Operators (FNOs), have rapidly advanced engineering practice by learning solution maps for complex PDEs with far greater efficiency and scalability than traditional solvers. Key real-world applications span seismic and acoustic wave propagation, multiphysics data assimilation, structural and materials modeling, multiphase/turbulent flow, and design optimization.

*Multiphysics and Data Assimilation.* Operator learning frameworks now serve as "plug-and-play" surrogates in coupled multiphysics systems. DeepM&Mnet [33] uses pre-trained DeepONets to enforce physics constraints alongside sparse measurements, achieving rapid electroconvection simulations for unseen potentials. SPI-MIONet [143] combines data-driven and physical constraints to improve hydraulic fracturing simulations, addressing challenges such as convergence and extensive data requirements. For high-speed boundary layers, a dual DeepONet scheme – forward and inverse – enables full data-assimilation cycles, reconstructing upstream disturbances from wall-pressure data [144]. In multiscale modeling contexts, DeepONet [102] is employed as a surrogate for high-fidelity solvers, allowing efficient bridging between coarse and fine resolutions. This enables practical and scalable simulation frameworks for coupled mechanical systems without sacrificing accuracy.

*Wave Propagation and Acoustic Modeling.* In geophysical seismic exploration and elastic wave simulation, operator architectures overcome the cost and stability limitations of grid-based methods. Multiscale Feature Extraction and Aggregation–Embedded FNO (MFEAFNet) [145] integrates attention and multiscale modules into FNO to capture fine elastic wave details in heterogeneous media. The Multiple-Input FNO (MIFNO) [146] handles 3D structured material properties and varying source parameters for accurate time-space wavefields, while the Factorized FNO (F-FNO) reduces memory and computation for large-scale 3D elastic simulations [147]. The boundary-element–augmented B-FNO [148] merges BEM integral formulations with FNOs for efficient exterior wave problems. FNO for Real-Time Microseismic Event Localization applies FNOs to microseismic monitoring, providing real-time, high-accuracy seismic event localization, essential for safe geothermal system development [149]. Moreover, En-DeepONet [73] extends DeepONet to moving-solution operators, delivering four orders of magnitude gains in accuracy for seismic hypocenter localization. FC-DeepONet [150] provides a more adaptable and accurate solution for seismic travel-time predictions, significantly improving the ability to predict travel-time fields in heterogeneous media. In acoustics, DeepONet-based surrogate [151] predicts millisecond-scale sound fields from moving sources in virtual rooms, enabling immersive audio experiences without precomputing impulse responses.

*Structural Mechanics and Materials Modeling.* In solid mechanics, neural operators blend physics knowledge with data-driven learning to predict complex material responses. The Implicit Fourier Neural Operator (IFNO) [104] formulates deep layers as integral operators, capturing nonlocal constitutive behavior for brittle and anisotropic materials, and outperforming classical constitutive models on digital image correlation data. A novel DeepONet with a ResUNet trunk [152] encodes intricate topology-optimized geometries and parametric loads to predict elasto-plastic stress fields orders of magnitude faster than finite-element analysis, while remaining memory-efficient. DeepONet for Thermo-Mechanical Analysis [153] enhances this domain further by combining ResUNet and sequential learning to predict thermal and mechanical stress fields in processes like steel casting and additive manufacturing. It achieves both higher speed and accuracy than traditional finite element analysis, enabling rapid industrial design and optimization. Peridynamic Neural Operator (PNO) [154] captures nonlocal material behaviors, offering high accuracy and robustness in material science applications, especially in noisy environments. The integration of finite element methods (FEM) with neural operators [155], as demonstrated in Physics-Informed Neural Operators for Micromechanics, ensures accurate predictions for stress and deformation in heterogeneous microstructures. This approach avoids relying on interpolation techniques, offering a more accurate solution for micromechanical problems compared to conventional methods. The Thermodynamically-informed Iterative Neural Operator (TherINO) [156] encodes constitutive thermodynamics to model elastic deformation in heterogeneous materials under periodic boundary conditions. Additionally, FNOs have been adapted to image classification tasks involving porous media [157], where the ability of FNOs to handle variable input sizes without architectural changes offers robustness for practical applications such as permeability prediction.

*Fluid Dynamics and Design Optimization.* For subsurface and reservoir engineering, U-FNO [103] extends FNO to $CO_2$–water multiphase flow, delivering accurate gas saturation and pressure-front predictions in highly heterogeneous porous media with only one-third the training data of CNN benchmarks. In high-Reynolds-number turbulence, an attention-enhanced neural operator [158] automatically focuses on nonequilibrium flow regions, reducing error by 40%with minimal parameter overhead, marking a significant leap in data-driven turbulence modeling. The Locality of Local Neural Operators (LNO) [159] explores the receptive range of LNOs for solving transient PDEs, revealing that performance is highly dependent on proper locality design. This work is particularly impactful in fluid dynamics, where accurately capturing spatiotemporal interactions can drastically improve the efficiency and accuracy of transient flow predictions. Neural operators have revolutionized parametric design by enabling rapid evaluation of full-field solutions across varying geometries. DeepONets infer flow around unseen airfoil shapes in hypersonic regimes [160], slashing optimization cost while preserving accuracy. Geom-DeepONet [161] encodes arbitrary 3D meshes via signed-distance functions and SIREN trunks to predict field solutions across dissimilar designs, accelerating preliminary design evaluation. The Boundary Assimilation FNO (BAFNO) [162] integrates parametric boundary conditions into the loss, improving generalization and speeding up flow-around-structure simulations without observational data. The Advanced Physics-Informed DeepONet for Aerospace [163] introduces architectural innovations including nonlinear decoders and curriculum learning, addressing the complexity of large, high-dimensional aerospace design spaces. It significantly improves predictive accuracy and generalization, extending applicability to broader nonlinear system design challenges.

Collectively, neural operators have proven to be a transformative tool in engineering, offering scalable, accurate, and efficient solutions for complex physical systems across various domains. From seismic and acoustic modeling to multiphase flow and material science, these methods are reshaping how we solve real-world engineering problems.

**Biomedical Applications of Neural Operators.** Neural operators are increasingly deployed in biomedical contexts to model complex physiological dynamics, enable rapid image-based diagnostics, and bridge macro–micro scales in tissue and cellular systems. Key innovations span cardiovascular mechanobiology, biochemical signaling, tissue mechanics, medical imaging, thrombosis modeling, and device design.

*Cardiovascular Mechanobiology and Remodeling.* Operator learning frameworks have been applied to predict disease progression in arterial pathologies and to assimilate personalized hemodynamics. DeepONet surrogate trained on finite-element–based growth–remodeling simulations [164] accurately infers thoracic aortic aneurysm risk factors from sparse or full-field distensibility maps, enabling patient-specific prognosis of aneurysm evolution. Similarly, DeepONet model of aortic dissection [165] uses phase-field finite-element data to predict pressure–volume curves and wall damage for varying microstructural strut distributions, offering a fast surrogate for in silico delamination studies. More recently, a comparative study of PINNs, DeepONets, and physics-informed DeepONets (PI-DeepONets) for 3D abdominal aortic aneurysm flow has demonstrated that operator-based surrogates can match CFD accuracy while dramatically reducing computational cost—highlighting their potential for real-time clinical decision support [166].

*Biochemical Modeling and Tissue Mechanics.* Modeling abrupt, nonlinear switches in cellular signaling poses challenges for traditional ODE solvers. The Multiscale Attention Wavelet Neural Operator (MAWNO) [167] combines wavelet transforms to capture multi-scale signal features with self-attention to emphasize high-frequency components, achieving superior accuracy on classical biochemical pathway models exhibiting steep dynamic changes. Neural operators have also been leveraged to learn constitutive behavior directly from experimental observations without predefined material models. A DeepONet [93] trained on digital image correlation (DIC) displacement data predicts porcine heart-valve leaflet deformation under novel loading scenarios, outperforming conventional FEA-based constitutive models by an order of magnitude in generalizability. Capturing both macro-scale flow and micro-scale platelet mechanics is critical for thrombosis research. DeepONet surrogate [168] trained on particle-dynamics simulations accurately predicts platelet membrane deformation under shear, integrating sub-cellular dynamics into larger-scale flow models with $<0.5\%$ error. Extending this, recent DeepONet [169] predicts both deformation and accumulated stress over time across spatial filtering radii, laying groundwork for efficient multi-scale thrombosis simulators.

*Biomedical Imaging and Device Design.* Neural operators have shown great promise in advancing both biomedical imaging and intelligent device optimization. In ultrasound computed tomography (USCT), the Neural Born Series Operator (NBSO) [170] accelerates full-wave forward simulations, enabling near–real-time image reconstruction pipelines for brain and breast imaging. Operator learning has also been applied to time-of-flight USCT, mapping TOF measurements directly to heterogeneous sound-speed fields in a single forward pass—first demonstrating real-time ultrasound tomography for soft-tissue tumor identification [171]. In elastography, a wavelet neural operator [17] extracts elasticity maps directly from noisy displacement fields, bypassing strain computation and delivering accurate tumor stiffness quantification in both synthetic and in vivo ultrasound data. Beyond modeling, neural operators are emerging in device optimization. A geometry-aware Fourier neural operator [172] was used to drive catheter tip designs that repel bacteria upstream swimming, reducing bacterial contamination by 1–2 orders of magnitude under clinical flow conditions—demonstrating how operator-based surrogates can inform AI-driven medical device geometries.

These examples underscore how neural operators – by learning mappings between high-dimensional functional spaces – are revolutionizing biomedical modeling, from cellular signaling to patient-specific cardiovascular predictions, real-time imaging, and device design, all with unprecedented speed and accuracy.

#### 3.3.6. Experiments

In this section, we have selected eight baseline datasets, with their fundamental information presented in Table 9 (for detailed PDE dataset descriptions, refer to Appendix A.3). Performance evaluation of the framework is conducted based on the $L_2$ relative errors predicted on dataset instances encompassing diverse features, while numerical experiments are executed on a single RTX 4090 GPU.

*Discussions:* Table 10 displays the results obtained from training diverse neural operators on datasets encompassing various challenges (Due to the lack of critical code, some experimental results we will show the author's optimal error for the sake of fairness). For general regular datasets like Darcy, GK-Transformer demonstrated the best performance, closely followed by OFormer with slightly lower accuracy in the second position. For the Naiver–Stokes dataset, FNO achieved the highest accuracy. Nevertheless, the training time of Transformer-based neural operators is much longer than that of FNO, etc., and FNOs can achieve similar accuracy. Therefore, in practical applications, neural operators represented by FNO often bring more cost-effective results for solving general regular datasets. Conversely, when dealing with complex datasets characterized by irregular grids and multiple scales (e.g., NACA, NS2d-c, and Inductor2d), GNOT outperformed other operators to a significant extent. In comparison to other neural operators, GNOT exhibits the ability to address multiple challenges presented by the dataset concurrently. Notably, the attention mechanism, serving as the central component of the transformer, empowers the model to capture dependencies and relationships among various elements within the sequence. However, this advantage comes at the expense of heightened computational requirements. Consequently, in order to effectively leverage the capabilities of attention-based models, it is imperative to possess an ample amount of training data to ensure robust performance and generalization across the designated task. Consequently, when dealing with general regular datasets such as Darcy flow, opting for neural operators represented by FNO can yield satisfactory accuracy within a relatively brief timeframe. Conversely, when confronted with intricate problems involving irregular grids, multiple scales, and similar challenges, transformer-based neural operators such as GNOT can proficiently exploit the progressively expanding datasets, resulting in decreased errors as the model capacity increases. This analysis aligns with the outcomes of numerical experiments. While transformer-based operators possess an advantage when operating on complex large datasets, their advantage diminishes when dealing with general regular datasets such as Darcy, where neural operators represented by FNO can deliver desirable outcomes.

## 4. Summary and outlook

Despite their limitations, neural operators continue to hold significant importance as a deep learning approach for solving PDEs and their associated applications. The table in Appendix C provides a summary of neural operators, along with their respective advantages and disadvantages. The development of neural operators can be broadly categorized into three directions: (1) exploring and extending basis functions to capture diverse features, (2) developing hybrid operators that integrate physical information and data, and (3) expanding operators to handle complex systems encountered in practical applications. Neural operators have exhibited exceptional performance in solving a range of PDEs, including Burgers' equation and Navier–Stokes equation, among

**Table 9**
Summary of experiment benchmarks.

| | Physics | Geometry | Dim | Feature |
|---|---|---|---|---|
| Darcy2d | Fluid | Regular grid | 2D | – |
| NS2d | Fluid | Regular grid | 2D | – |
| Elasticity | Fluid | Structured mesh | 2D | Irregular |
| NS2d-c | Fluid | Structured mesh | 2D | Irregular; multi-scale |
| Airfoil | Gas | Structured mesh | 2D | Irregular; multi-scale |
| Inductor2d | Electromagnetic | Structured mesh | 2D | Irregular; multi-scale |
| Heat | Fluid; gas | Structured mesh | 2D | Irregular; multiple input; multi-scale |

**Table 10**
Results of the operator method trained on multiple datasets, with the best results shown in bold, and the second-best results are shown underlined.

| | | MIONet (%) | FNO (%) | GK-Transformer (%) | Geo-FNO (%) | U-FNO (%) | OFormer (%) | GNOT (%) |
|---|---|---|---|---|---|---|---|---|
| Darcy2d | – | 2.98 | <u>1.35</u> | **0.84** | 1.35 | 1.65 | 1.28 | 2.01 |
| NS2d | – | – | **0.26** | 1.20 | 0.26 | 0.45 | <u>0.42</u> | 0.65 |
| Elasticity | – | 9.65 | – | 2.01 | 2.20 | – | <u>1.83</u> | **0.87** |
| | $u$ | 2.74 | – | 6.56 | <u>1.41</u> | – | 2.33 | **0.67** |
| NS2d-c | $p$ | 5.51 | – | 11.50 | <u>2.98</u> | – | 4.83 | **1.55** |
| | $v$ | 2.74 | – | <u>1.11</u> | 1.62 | – | 2.43 | **0.74** |
| NACA | – | 13.20 | – | 1.61 | <u>1.38</u> | – | 1.83 | **0.76** |
| | $A_z$ | 3.10 | – | 25.60 | – | – | <u>2.23</u> | **1.21** |
| Inductor2d | $B_x$ | 3.49 | – | 3.06 | – | – | <u>2.83</u> | **1.92** |
| | $B_y$ | 6.73 | – | 4.45 | – | – | <u>4.28</u> | **3.62** |
| Heat | – | 14.50 | – | – | – | – | – | **2.56** |

others. These methods have applications in diverse fields, encompassing physics, engineering, and uncertainty quantification [173,174]. Despite the introduction of numerous novel neural operators, the speed of FNO and the flexibility of DeepONet have positioned them as the most extensively utilized approaches. Furthermore, the versatility and broad applicability of neural operators make them invaluable tools in various scientific and engineering domains. From the perspective of tasks of neural operators, we also suggest several research directions and opportunities:

*Selection of operator basis:* Based on the previous discussion, it is evident that the selection of the operator basis significantly influences the performance of neural operators. For instance, the utilization of PR-LNO enables the capture of transient responses in the data, whereas the wavelet neural operator effectively learns spatial changes in equations by capturing variations in spatial frequencies. Hence, the selection and transformation of basis functions in neural operators continue to be subjects of ongoing discussion. For example, can the selection of distinct wavelet families in wavelet basis functions enhance the efficacy of neural operators in solving specific types of equations? Is it possible to incorporate commonly utilized radial basis functions employed in solving PDEs into neural operators? These avenues of investigation present intriguing prospects for future exploration.

*Physical information and data-driven:* The combination of physics constraints and data-driven operators represents a highly esteemed and promising avenue of research. When dealing with some problems, PINNs may struggle to learn the accurate solutions even in simple cases [175]. Data-driven neural networks necessitate extensive and costly datasets comprising precise training data for extracting and learning the underlying physical laws. This circumstance has spurred the development of hybrid neural operator approaches, which leverage the benefits of physical models and large datasets to enhance accuracy, interpretability, and predictive capabilities. Consequently, these approaches drive progress in pertinent fields and address real-world problems. Nevertheless, hybrid neural operators encounter several challenges in practical implementation. For instance, operator learning might fall short in generating predictions consistent with the underlying physical laws, making the effective integration of information from both methods an intriguing area of research. Moreover, the ill-conditioned nature of the loss in PINNs, attributed to the residual loss [176], renders the attainment of the optimal solution challenging, thereby underscoring the criticality of selecting an appropriate optimizer. Nevertheless, a paucity of discussion and exploration exists concerning the selection of optimizers in hybrid neural operators. Consequently, the development of optimizers explicitly tailored to individual neural operators and the proposal of more precise derivative methods for PDE constraints continue to represent promising avenues of research.

*Data distribution and sampling:* Existing operator methods, whose training and test data are usually sampled from Gaussian processes with custom covariance kernels, may be impractical for real-world complex problems and applications. A significant problem is to study the impact of the distribution of data on the performance of neural operators, both from a practical and theoretical viewpoint. Researchers have introduced multiple physics pretraining (MPP) to predict the dynamics of multiple heterogeneous physical systems simultaneously by learning features that are broadly useful across diverse physical tasks [177]. Designing a large, generic, transferable, generalized neural operator framework is also a major challenge.

*Embedding of existing models and frameworks:* Neural operators have a relatively short history, and the existing models and frameworks are fairly concentrated and uniform. It is also a very valuable research direction to introduce classical numerical methods and popular and powerful models from the field of machine learning. A hybrid iterative method based on MIONet for PDEs has been proposed, which combines the traditional numerical iterative solver and the recent powerful machine learning method of neural operator [178]. Diffusion transformer architecture, Vision Transformer and State-space models have been introduced into operator learning [179–181]. Embedding models and frameworks from computer vision, natural language processing, and other fields into existing neural operators helps improve computational efficiency and modeling capabilities to address the limitations of existing architectures.

*Software and datasets:* DeepXDE [30] presents a Python library for PINNs and PINNacle [182] introduces a benchmarking tool designed to make a comprehensive comparison of these methods across a wide range of Partial Differential Equations (PDEs). As yet, a widely accessible, practically simple, and statistically challenging benchmark with ready-to-use datasets to compare methods in Neural Operator is missing. Establishing a list of standard PDE problems across different scientific fields, such as heat conduction, fluid dynamics, biology, and electromagnetics, with other properties (e.g. linear/nonlinear, steady/time-dependent, low/high-dimensional, smooth/rough solution, simple/complex geometry) would allow researchers to compare and identify the neural operator architectures that are the most appropriate for a particular task.

*New application directions:* Compared to traditional neural networks that learn point-to-point mappings, neural operators directly learn the mapping relationship between function spaces. Consequently, neural operators demonstrate superior generalization performance and can readily be applied to a wide range of problems. In climate assessment and prediction models, which involve large-scale system equation modeling, neural operators can efficiently parallelize computations on a large scale, mitigating the computational cost associated with repetitive training. The direct learning of mappings between functions and the unique discretization invariance of neural operators hold significant potential for evaluating and predicting various factors in climate studies, encompassing precipitation, temperature, and other influential parameters. Moreover, in the field of computer vision, images and 3D models can be treated as functions within the spatial domain, rendering the discretization invariance advantages of neural operators a viable and exceptionally promising avenue for future exploration. For instance, researchers have successfully expanded the application of FNO to image super-resolution, yielding satisfactory outcomes [24]. Neural operators can find applications in diverse vision domains, including image synthesis and video super-resolution, among others.

*Theory:* Extensive theoretical analysis is essential to gain a profound understanding and effectively address the inherent challenging problems of neural operators. The approximation theory of many neural operators has been proven in recent studies [59,183–188]. However, the universal approximation serves as a necessary but insufficient condition for the success of the architecture, as it does not guarantee the attainment of the theoretically expected accuracy in practical model applications [189]. Therefore, research on approximation theory plays a pivotal role in enhancing the accuracy, stability, and reliability of neural operators. Furthermore, in neural operator research, it is crucial to explore the theoretical foundations for network architecture selection and investigate interpretability. For instance, the experiments conducted in this paper reveal that solving the Advection equation iteratively with time steps yields unsatisfactory results. However, greater accuracy can be attained by training the model while treating time as an independent dimension. In fact, given the distinctive characteristics and properties of various types of PDEs, investigating the theoretical foundations of different operators that are suitable for solving the corresponding equations represents a crucial research direction. Through a comprehensive examination of the general approximation theory and interpretability of neural operators, their applicability in diverse fields and applications can be ascertained, thereby facilitating their widespread adoption in scientific computing and model solving.

## CRediT authorship contribution statement

**Shengjun Liu:** Writing – review & editing, Writing – original draft, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Yu Yu:** Writing – review & editing, Writing – original draft, Software, Resources, Methodology, Data curation, Conceptualization. **Ting Zhang:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization. **Hanchao Liu:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Xinru Liu:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Data curation. **Deyu Meng:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Data curation.

## Declaration of competing interest

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.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62172447), Hunan Provincial Innovation Foundation for Postgraduate, China (No. CX20240166), and the Graduate Student Research and Innovation Program of Central South University (No. 2024ZZTS0729). And the authors would like to thank technical support from the High Performance Computing Center of Central South University.

## Appendix A. Details of datasets

### A.1. Extension based on operator basis

*1D Burgers' equation:* The one-dimensional Burgers' equation is a type of nonlinear PDEs that is often used to simulate the one-dimensional flow of viscous liquids and gases. Its expression is as follows,

$$\begin{aligned} \partial_t u(x,t) + \partial_x \left(u^2(x,t)/2\right) &= \nu \partial_{xx} u(x,t), \quad && x \in (0,1), t \in (0,1], \\ u(x,0) &= u_0(x), && x \in (0,1), \end{aligned} \tag{A.1}$$

which has periodic boundary conditions: $u(x=0,t) = u(x=1,t)$, $u_0 \in L^2_{\text{per}}((0,1);\mathbb{R})$ is the initial condition, and $\nu \in \mathbb{R}_+$ is the viscosity factor. The goal is to learn the operator: $u_0 \mapsto u(\cdot,1)$. The dataset is from Ref. [19] with viscosity coefficient $\nu = 0.1$ and spatial resolution of 1024.

*Burgers' equation with discontinuity:* For smaller values of the viscosity parameter $\upsilon$, the Burgers' equation creates sharp jumps in the velocity field [17]. Such discontinuous Burgers' equations are expressed as follows:

$$\begin{aligned} \partial_t u(x,t) + u(x,t)\partial_x u(x,t) &= \frac{0.01}{\pi} \partial_{xx} u(x,t), && x \in [-1,1], t \in [0,1], \\ u(x=-1,t) &= u(x=1,t) = 0, && x \in [-1,1], t \in [0,1], \\ u(x,0) &= -\sin(\pi x) + \zeta \sin(\pi x), && x \in [-1,1], \end{aligned} \tag{A.2}$$

where $\zeta \in \mathbb{R}$ is randomly selected from $[0, 0.5]$. The aim is to learn the operator: $u|_{(-1,1)\times[0,10]} \mapsto u|_{(-1,1)\times[10,50]}$. The dataset comes from the Ref. [17], time step is $\Delta t = 0.02$, and spatial resolution is $512 \times 512$.

*1D time-dependent wave advection equation:* The one-dimensional time-dependent wave advection equation is a hyperbolic PDE, which is mainly used to describe the solution of a scalar with a discontinuity in some known velocity field. The advection equation with periodic boundary conditions is represented as follows,

$$\begin{aligned} \partial_t u(x,t) + \upsilon \partial_x u(x,t) &= 0, && x \in (0,1), t \in (0,1), \\ u(x-\pi,t) &= u(x+\pi,t), && x \in (0,1), t \in (0,1), \end{aligned} \tag{A.3}$$

where $\upsilon \in \mathbb{R}^{+*}$ represents the flow rate with periodic boundary conditions $u(x,0) = h1_{\left\{c-\frac{\omega}{2}, c+\frac{\omega}{2}\right\}} + \sqrt{\max\left(h^2 - (a(x-c))^2, 0\right)}$, $(c, w, h)$ are randomly chosen from $[0.3, 0.7]\times[0.3, 0.6]\times[1, 2]$. The goal is to learn the operator: $u|_{(0,1)\times[0,0.025]} \mapsto u|_{(0,1)\times[0,1]}$. The dataset is from Ref. [64], $\upsilon = 1$, and has a spatial resolution of $40 \times 40$.

*2D Darcy flow equation:* The two-dimensional Darcy flow equations are widely used to model the flow through porous media, and in the two-dimensional spatial domain they are expressed as:

$$\begin{aligned} -\nabla \cdot (a(x)\nabla u(x)) &= f(x), && x \in (0,1)^2, \\ u(x) &= 0, && x \in \partial(0,1)^2, \end{aligned} \tag{A.4}$$

where $u(x,y) = u_0(x,y)$ is the initial condition, $a(x,y)$ is the permeability field, $u(x,y)$ is the pressure, $\nabla u(x,y)$ is the pressure gradient, and $f(x,y)$ is the source function. The goal is to learn the operator: $a(x,y) \mapsto u(x,y)$. The dataset is from Ref. [19], with initial conditions $u_0(x,y) = 0$, $f(x,y) = 1$, and has a spatial resolution of $85 \times 85$.

*2D time-dependent Navier–Stokes equation:* The Navier–Stokes equations are two-dimensional time-dependent PDEs used to model the motion of nonlinear incompressible viscous fluids. The representation is as follows,

$$\begin{aligned} \partial_t w(x,t) + u(x,t)\cdot\nabla w(x,t) &= \nu\Delta w(x,t) + f(x), && x\in(0,1)^2, t\in(0,T], \\ \nabla\cdot u(x,t) &= 0, && x\in(0,1)^2, t\in[0,T], \\ w(x,0) &= w_0(x), && x\in(0,1)^2, \end{aligned} \tag{A.5}$$

where $v\in\mathbb{R}, f(x,y)$ represents the viscosity of the fluid and the source function, and $u(x,y,t), w(x,y,t)$ are the velocity and vorticity fields of the fluid, respectively. The goal is to learn: $w|_{(0,1)^2\times[0,10]}\mapsto w|_{(0,1)^2\times[10,20]}$. The dataset is from Ref. [19], with a viscosity coefficient $\nu = 0.001$, initial conditions $f(x,y) = 0.1(sin(2\pi(x+y)) + cos(2\pi(x+y)))$, and has a spatial resolution of $64\times 64$.

*2D Allen-Cahn equation:* The Allen-Cahn equation is a reaction–diffusion equation with a nonlinear reaction term. The Allen-Cahn equation in two dimensions is given by the following equation:

$$\begin{aligned} \partial_t u(x,y,t) &= \epsilon\Delta u(x,y,t) + u(x,y,t) - u(x,y,t)^3, && x,y\in(0,3), t\in[0,20], \\ u(x=0,y,t) &= u(x=3,y,t), && y\in(0,3), t\in[0,20], \\ u(x,y=0,t) &= u(x,y=3,t), && x\in(0,3), t\in[0,20], \\ u(x,y,0) &= u_0(x,y) && x,y\in(0,3). \end{aligned} \tag{A.6}$$

Its initial condition uses $\mathcal{K}(x,y) = \tau^{(\alpha-1)}\left(\pi^2\left(x^2+y^2\right)+\tau^2\right)^{\frac{\alpha}{2}}$. The kernel is simulated from a Gaussian random field: $\tau = 15, \alpha = 1$. The goal is to learn the operator: $u_0(x,y)\mapsto u(x,y,t)$, t = 20 s. The dataset comes from the Ref. [17], the time step is $\Delta t = 0.02$, and the spatial resolution is $43\times 43$.

*Euler's Equation(Airfoil):* Consider transonic flow over an airfoil with complex boundary conditions, where the governing equations are two-dimensional nonlinear Euler equations:

$$\begin{aligned} &\frac{\partial\rho^f}{\partial t} + \nabla\cdot\left(\rho^f\boldsymbol{v}\right) = 0, \\ &\frac{\partial\rho^f\boldsymbol{u}}{\partial t} + \nabla\cdot\left(\rho^f\boldsymbol{v}\otimes\boldsymbol{v} + p\mathbb{I}\right) = 0, \\ &\frac{\partial E}{\partial t} + \nabla\cdot((E+p)\boldsymbol{v}) = 0, \end{aligned} \tag{A.7}$$

where $\rho^f$ is the fluid density, $v$ is the velocity vector, $p$ is the pressure, and $E$ is the total energy. The goal is to learn the operator: $(x,y)\mapsto u(x,y)$. The dataset is derived from Ref. [98] and is given as a point cloud of size approximately 1000.

*Hyper-elastic material(Elasticity):* The governing equation for a solid elastic material is given by,

$$\rho^s\frac{\partial^2\boldsymbol{u}}{\partial t^2} + \nabla\cdot\sigma = 0, \tag{A.8}$$

where $\rho^s$ is the mass density, $\boldsymbol{u}$ is the displacement vector, and $\boldsymbol{\sigma}$ is the stress tensor. The goal is to learn the operator: $(x,y)\mapsto\boldsymbol{\sigma}$. The dataset is from Ref. [98].

### A.2. Physical information and data-driven

*1D Burgers' equation:* This experiment uses the same nonlinear Burgers' equation with periodic boundary conditions as in Section 3.1, taking the viscosity coefficient $\nu = 0.01$, the spatial resolution is 128, and the temporal resolution is 101. The learning objective of this experiment is to learn the operator that maps the initial conditions to the solution: $u_0\mapsto u|_{[0,1]}$.To test the Ex.+, the Burgers' equation correlation length of train datasets is $l_{train} = 1.0$, and the test dataset of 100 functions with $l_{test} = 0.6$.

*2D advection equation:* To study the performance of physically based operator methods in dealing with partial differential equations dominated by advection, a linear advection equation with variable coefficients is considered:

$$\begin{aligned} &\frac{\partial s}{\partial t} + u(x)\frac{\partial s}{\partial x} = 0, (x,t)\in(0,1)\times(0,1), \\ &s(x,0) = f(x), \\ &s(0,t) = g(t), \end{aligned} \tag{A.9}$$

where $f(x) = sin(\pi x), g(x) = sin(\frac{\pi}{2}t)$. The spatial and temporal resolutions are both 100. The learning objective of this operator is to learn the mapping of variable coefficients to solutions: $u(x)\mapsto s(x,t)|_{(0,1)\times(0,1)}$. To test the Ex.+, the $u(x)$ is randomly sampled from a GRF with an RBF kernel with the correlation length $l_{train} = 0.5$, and the test dataset of 100 functions with $l_{test} = 0.2$.

*2D Diffusion-reaction dynamics:* This example tests an implicit operator with a nonlinear reaction–diffusion term:

$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), (x,t)\in(0,1]\times(0,1], \tag{A.10}$$

where the initial boundary condition is 0, the diffusion coefficient $D = 0.01$, and the reaction rate $k = 0.01$. The spatial and temporal resolution are both 100. The learning objective of this operator is to learn the mapping from the source term to the solution: $u(x)\mapsto s(x,t)|_{(0,1]\times(0,1]}$. To test the Ex.+, the $u(x)$ is randomly sampled from a GRF with an RBF kernel with the correlation length $l_{train} = 0.5$, and the test dataset of 100 functions with $l_{test} = 0.2$.

*3D Allen-Cahn dynamics:* Finally, to test the high-dimensional case, we consider the Allen-Cahn equation:

$$\begin{aligned} \partial_t u(x,y,t) &= \epsilon\Delta u(x,y,t) + u(x,y,t) - u(x,y,t)^3, && x,y,t\in[0,1], \\ u(x,y,0) &= u_0(x,y), && x,y\in[0,1], \end{aligned} \tag{A.11}$$

where the viscosity coefficient $\epsilon = 0.01$, has periodic boundary conditions. The spatial resolution is $65\times 65$ and the time step is 0.02. The goal of this study is to learn the operator: $u(x,y,t)|_{[0,1]^2\times[0,10]}\mapsto u(x,y,t)|_{[0,1]^2\times[10,20]}$.

### A.3. Complex systems

*Darcy flow/Naiver–Stokes/Elasticity/NACA(airfoil) equation:* The two-dimensional Darcy equation, Naiver–Stokes equation, NACA (airfoil) equation and Elasticity equation are used in this section, and their parameters are consistent with Appendix A.1.

*2D NS2d-c equation:* This example is the two-dimensional steady-state Naiver–Stokes equation defined on a rectangle minus four circular areas. The equation form is:

$$\begin{aligned} (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} &= \frac{1}{\mathrm{Re}}\nabla^2\boldsymbol{u} - \nabla p, \\ \nabla\cdot\boldsymbol{u} &= 0. \end{aligned} \tag{A.12}$$

The domain $\Omega = [0,8]^2\setminus\bigcup_{i=1}^4 R_i$, $R_i$ is a circle. The data set comes from [31].

*2D Inductor equation:* The two-dimensional inductor satisfying the following steady-state Maxwell's equations is represented as follows:

$$\begin{aligned} \nabla\times\boldsymbol{H} &= \boldsymbol{J}, \\ \boldsymbol{B} &= \nabla\times\boldsymbol{A}, \\ \boldsymbol{J} &= \sigma\boldsymbol{E} + \sigma\boldsymbol{v}\times\boldsymbol{B} + \boldsymbol{J}_e, \\ \boldsymbol{B} &= \mu_0\mu_r\boldsymbol{H}. \end{aligned} \tag{A.13}$$

It satisfies the boundary condition $n\times A = 0$. The dataset is derived from [31].

*2D Heat equation and 3D Heatsink equation:* An expression satisfying the 2d steady state heat equation is given below:

$$\rho C_p\boldsymbol{u}\cdot\nabla T - k\nabla^2 T = Q. \tag{A.14}$$

Its defining domain is $\Omega = [0,9]^2$ and it satisfies periodic boundary conditions on the left and right boundaries. And the three-dimensional

**Table B.11**
FNO, F-FNO and WNO architectures for each problem.

| | FNO(epochs = 500) | | F-FNO(epochs = 500) | | WNO(epochs = 500) | | |
|---|---|---|---|---|---|---|---|
| | Modes | Width | Modes | Width | Level | Width | Wavelet |
| 3.1.4 Burgers'BC | 16 | 64 | 16 | 64 | 8 | 64 | db6 |
| 3.1.4 Burgers' discontinuity | 16 | 32 | 16 | 32 | 6 | 40 | db6 |
| 3.1.4 1D time_advection | 16 | 32 | 16 | 32 | 3 | 80 | db6 |
| 3.1.4 Advection | 16 | 32 | 16 | 32 | 3 | 96 | db6 |
| 3.1.4 Darcy rectangular | 16 | 32 | 16 | 32 | 4 | 64 | db6 |
| 3.1.4 NS2d_time | 12 | 32 | 12 | 32 | 4 | 30 | db6 |
| 3.1.4 NS3d | 8 | 32 | 8 | 32 | 2 | 40 | db6 |
| 3.1.4 AC_BC2d | 16 | 32 | 16 | 32 | 1 | 64 | cwt |
| 3.1.4 Airfoil | 12 | 32 | 12 | 32 | 4 | 64 | db6 |
| 3.1.4 Elasticity | 16 | 32 | 16 | 32 | 3 | 64 | db6 |

**Table B.12**
Waveformer, LSM and L-DeepONet architectures for each problem.

| | Waveformer | | | LSM(epochs = 500) | | | L-DeepONet | | |
|---|---|---|---|---|---|---|---|---|---|
| | Epochs | Level | Width | Basis | Token | Width | Encode | latent_dim | Epochs |
| 3.1.4 Burgers'BC | – | – | – | 12 | 4 | 64 | – | – | – |
| 3.1.4 Burgers' discontinuity | 120 | 4 | 64 | 12 | 4 | 64 | MLAE | 144 | [2000,50 000] |
| 3.1.4 1D time_advection | – | – | – | 12 | 4 | 64 | MLAE | 64 | [6000,50 000] |
| 3.1.4 Advection | – | – | – | 12 | 4 | 64 | – | – | – |
| 3.1.4 Darcy rectangular | – | – | – | 12 | 4 | 64 | – | – | – |
| 3.1.4 NS2d_time | 300 | 4 | 28 | 12 | 4 | 64 | MLAE | 64 | [6000,50 000] |
| 3.1.4 NS3d | – | – | – | 12 | 4 | 64 | – | – | – |
| 3.1.4 AC_BC2d | – | – | – | 12 | 4 | 64 | – | – | – |
| 3.1.4 Airfoil | – | – | – | 12 | 4 | 32 | – | – | – |
| 3.1.4 Elasticity | – | – | – | 12 | 4 | 32 | – | – | – |

**Table B.13**
DeepONet and POD-DeepONet architectures for each problem.

| | DeepONet | | POD-DeepONet | |
|---|---|---|---|---|
| | Branch net | Trunk net | Branch net | POD |
| 3.1.4 Burgers'BC | Depth 4 & Width 128 | Depth 4 & Width 128 | Depth 3 & Width 128 | 32 |
| 3.1.4 Burgers' discontinuity | Depth 4 & Width 128 | Depth 3 & Width 128 | Depth 2 & Width 128 | 32 |
| 3.1.4 1D time_advection | – | – | – | – |
| 3.1.4 Advection | Depth 2 Width 512 | Depth 4 & Width 512 | Depth 2 & Width 512 | 32 |
| 3.1.4 Darcy rectangular | CNN | Depth 4 & Width 128 | CNN | 10 |
| 3.1.4 NS2d_time | – | – | – | – |
| 3.1.4 NS3d | CNN | [128, 128, 64] | CNN | 29 |
| 3.1.4 AC_BC2d | CNN | Depth 4 & Width 128 | CNN | 225 |
| 3.1.4 Airfoil | Depth 4 & nbsp; & nbsp; Width 128 | Depth 3 & Width 128 | CNN | 215 |
| 3.1.4 Elasticity | Depth 4 & Width 128 | Depth 4 & Width 128 | Depth 2 & Width 512 | 40 |

**Table B.14**
SNO, LNO and NOMAD architectures for each problem.

| | SNO | | LNO | | | NOMAD | |
|---|---|---|---|---|---|---|---|
| | Width | Epochs | Epochs | Modes | Width | Branch net & Trunk net | n |
| 3.1.4 Burgers'BC | 10 | 50 000 | 1000 | 16 | 4 | Depth 2 & Width 128 | 128 |
| 3.1.4 Burgers' discontinuity | 10 | 50 000 | 1000 | 16 | 4 | Depth 5 & Width 100 | 10 |
| 3.1.4 1D time_advection | – | – | 1000 | 16 | 4 | – | – |
| 3.1.4 Advection | 10 | 50 000 | 1000 | 4 | 48 | Depth 5 & Width 100 | 10 |
| 3.1.4 Darcy rectangular | – | – | 1000 | 4 | 16 | Depth 5 & Width 100 | 20 |
| 3.1.4 NS2d_time | – | – | 1000 | 4 | 16 | – | – |
| 3.1.4 NS3d | 10 | 30 000 | 500 | 4 | 8 | Depth 4 & Width 256 | 12 |
| 3.1.4 AC_BC2d | 10 | 50 000 | 1000 | 4 | 48 | Depth 5 & Width 512 | 10 |
| 3.1.4 Airfoil | 10 | 30 000 | 1000 | 4 | 16 | Depth 5 & Width 100 | 10 |
| 3.1.4 Elasticity | 10 | 30 000 | 1000 | 4 | 16 | Depth 4 & Width 256 | 20 |

Heatsink equation fluid satisfies the Naiver–Stokes equation and the heat equation. The flow and temperature fields are coupled by thermal convection and heat conduction. The dataset is obtained from [31].

## Appendix B. Parameters of experiments

See Tables B.11–B.15.

## Appendix C. Summary of nos.

See Table C.16.

## Data availability

Data sources can be seen in the Appendix A. The source codes for reproducing the results presented in this paper can be accessed at https://github.com/YuYu8900/Neural-Operators.

**Table B.15**
FNO, WNO and DeepONet architectures for 3.2.3 each problem.

| | FNO | | WNO | | | DeepONet | |
|---|---|---|---|---|---|---|---|
| | Modes | Width | Level | Width | Wavelet | Branch net | Trunk net |
| 3.2.3 Burger's | 12 | 32 | 4 | 32 | db6 | Depth 7 & Width 100 | Depth 7 & Width 100 |
| 3.2.3 Advection | 16 | 32 | 4 | 32 | db6 | Depth 6 & Width 100 | Depth 6 & Width 100 |
| 3.2.3 Diffusion_reaction | 16 | 64 | 4 | 32 | db6 | Depth 5 & Width 50 | Depth 6 & Width 100 |
| 3.2.3 Allen-Cahn3D | 16 | 32 | 4 | 32 | db4 | Depth 7 & Width 100 | Depth 7 & Width 100 |

**Table C.16**
Summary of neural operators.

| Class | NO structure | Originality | Advantage | Disadvantage |
|---|---|---|---|---|
| Vanilla | DeepONet [16] | Neural operators were proposed for the first time. | Efficiently learn operator mapping and achieve good accuracy in solving PDEs. | Lack of discretization invariance. |
| | GNO [20] | Introducing graph networks into operator learning. | Excelling at handling inputs in graph format. | Taking up a great deal of memory and runs slowly. |
| | FNO [19] | Proposing an alternative neural operator architecture defined directly in Fourier space. | Obtaining satisfactory accuracy while solve the problem quickly. | Restricted to regular grid. |
| | GK-Transformer [21] | Introducing transformer into the neural operator. | Achieving significant improvements in complex PDEs. | The input and output features of the embedding are defined on the same mesh. |
| Basis function | MGNO [37] | Decomposing the operator kernel into multi-level subkernels. | Capturing short-range to long-range interactions. | Taking up a great deal of memory and runs slowly. |
| | MWT [46] | Introducing multi-wavelet filters into the integral operator kernel. | Enhancing the robustness and accuracy for high-dimensional PDEs. | Only applicable to inputs with Quadratic resolution. |
| | WNO [17] | Introducing wavelet transform into the integral operator kernel. | More effectively capturing spatial features and overcoming the limitation of regular grids. | Theoretical results regarding network approximation of operators. |
| | W-DeepONet [47] | Introducing discrete wavelet transform into DeepONet to effectively handle spatial information. | Effectively capturing transient and non-transient responses. | Applied research on different types of equation problems. |
| | PR-LNO [48] | Introducing Laplace transform into the integral operator kernel. | Capable of handling non-periodic signals and better capturing transient responses. | Relevant theoretical results and specific applications in various directions. |
| | SNO [51] | Introducing trigonometric basis and Chebyshev basis into the integral operator kernel. | Reducing aliasing errors while overcoming the limitations of regular grids. | Only applicable to smooth input and output data. |
| | LOCA [58] | Proposing a neural operator based on novel RKHS Tikhonov regularization, providing a theory. | Providing a rigorous identifiability analysis and convergence study for non-local operators. | Specific practical applications. |
| | NNO [60] | Providing a new universal approximation theorem as a theoretical foundation that is not constrained by geometric shapes. | Provide a general and universal theory that transcends geometric shape constraints. | Specific practical applications. |
| | NKN [44] | Proposing a combination of GNO and nonlocal neural network integral kernel forms. | Improving stability and generalization capabilities. | Research on high-dimensional problems. |
| Basis decomposition | F-FNO [61] | Introducing Fourier factorization, allowing for independent processing of each spatial dimension. | Improving operator stability and reducing model complexity. | Theoretical research on whether the approximation theory hold under Fourier factorization. |
| | POD-DeepONet [64] | Introducing POD into DeepONet to pre-learn operator basis functions. | Reducing the dimensionality of the data and improving performance. | Research and application studies on high-dimensional complex problems. |
| | SSNO [65] | The features obtained through filtering decomposition are fused into the Fourier layer. | Improving the performance in the spatial domain. | Theory regarding the network size for approximating operators. |

(*continued on next page*)

**Table C.16** (*continued*).

| Class | NO structure | Originality | Advantage | Disadvantage |
|---|---|---|---|---|
| Basis space | Improved-DeepONet [67] | Using an encoder to embed inputs into high-dimensional feature space, and merge information at each layer. | Enhancing its ability to represent nonlinearity and preserve the input signal. | Theoretical research on the approximation properties of model frameworks. |
| | LSM [69] | Using Attention mechanisms to project input into latent space and selecting triangular basis as the operator basis. | Highlighting the properties of data and overcome geometric shape limitations. | The design of the solution process is relatively complex. |
| | L-DeepONet [70] | Using unsupervised autoencoder to project data into a latent space | Making it easier to capture feature information. | Restricted to solving time-dependent PDEs. |
| | Oommen et al. [71] | Projecting to the latent space to accelerate the prediction of microstructure evolution. | Removing redundant information to capture the main features. | Research on the application of different PEDs types. |
| Physics-Informed | PI-DeepONet [76] | Building a physics-informed DeepONet model under pure physical information conditions. | Overcoming data constraints to enhance generalization capabilities. | The predictive accuracy may decrease with steep gradients in PDE solutions. |
| | PINO [77] | Introducing physical information into FNO training and developing an accurate derivative method. | Enhancing the across-resolution generalization capabilities to overcome data limitations. | Theoretical foundation of derivative methods still requires further improvement. |
| | PI-WNO [78] | Building physics-informed WNO and introducing a derivative mechanism based on random projections. | Improving performance and effectively learning the solution without training data. | The derivative calculation speed is relatively slow. |
| | Pi-V-DeepONet [80] | Proposing a physics-informed DeepONet variational formulation for brittle fracture mechanics. | Effectively learning the solution operator without training data. | Research on the application of different PDEs problems. |
| | FC_PINO [83] | Leveraging Fourier continuation to apply accurate gradient methods to non-periodic problems. | Achieving more accurate derivative calculation. | Specific application practices. |
| Physics-Informed Arch.&App. | Multifidelity-DeepONet [85,86] | Proposing a multi-fidelity DeepONet approach that significantly reduces the high-fidelity data. | Significantly reducing the required amount of high-fidelity data while achieving small-scale errors. | There is room for improvement in the training process. |
| | DeepM&Mnet [90] | Proposing a framework that can construct complex models using only small data and partial physical information. | Achieving good accuracy even in the presence of scarce data. | It is necessary to pre-train DeepONets models with sufficient data beforehand. |
| | DeepM&Mnet [33,90] | Using pre-trained DeepONets to construct complex models based on extremely limited data. | Achieving good accuracy in the case of sparse data, demonstrating generalizability. | It is important to pre-train the model with sufficient data beforehand. |
| | Extrapolation DeepONet [92] | Providing a general approach for extrapolation and propose several methods to enhance the extrapolation performance. | Effectively improving the extrapolation performance of DeepONet. | Theoretical research on model error estimation. |
| Irregular | gFNO [64] | Extending FNO by extending the domain to handle mappings defined on complex geometries. | Breaking free from geometric shape limitations in the input domain. | May introduce additional errors. |
| | Geo-FNO [98] | Generalizing FNO to solve PDEs on arbitrary shapes while preserving the speed. | Overcoming geometric shape limitations while maintaining the original speed. | Theoretical research on model errors. |
| | Geo-LNO [49] | Proposing LNO which breaks the grid limitations of FNO while maintaining discretization invariance. | Breaking the grid limitations of FNO while preserving discretization invariance. | Theoretical support for convergence. |
| | BelNet [18] | Combining the strengths of DeepONet and FNO to propose a mesh-free operator. | Enabling arbitrary irregular inputs and outputs while maintaining discretization invariance. | Generalization proofs for the extension of universal approximation theorems. |
| | OFormer [99] | Constructing a new operator architecture which breaks the grid limitations based on transformers. | Handling varying numbers of input points and process both uniform and non-uniform discretized grids. | Sufficient data is required to achieve optimal performance. |

(*continued on next page*)

**Table C.16** (*continued*).

| Class | NO structure | Originality | Advantage | Disadvantage |
|---|---|---|---|---|
| Multi-scale | Multiscale DeepONet [101] | Applying multiscale DNNs separately to DeepONet to solve multiscale problems. | Expanding DeepONet to address multiscale problems. | Research and studies on dynamic nonlinear problems. |
| | Yin et.al [102] | Using concurrent coupling methods to couple neural operators with traditional numerical solvers. | Significantly reducing the computational costs. | Lack of performance evaluation for long-term predictions. |
| | MNO [105] | Utilizing known physical information to simulate large-scale dynamics. | Significantly reducing computational costs and improving stability. | Not independent in time. |
| | U-FNO [103] | Adding a two-step U-Net within the Fourier domain to apply in multiphase flow. | Reducing the required amount of data and memory requirements. | Sacrifices discretization invariance. |
| | HT-Net [106] | Using hierarchical discretization to enhance the model's ability to capture the oscillatory characteristics. | Having better accuracy and improved generalization capabilities. | Tailored for regular or structured grids. |
| | IFNO [104] | Applying the representation of the solution operator as an implicitly defined mapping to multiscale problems. | Improving the accuracy and stability for solving multiscale problems while reducing memory requirements. | Increasing computational costs and limited applicability to structured grids. |
| Multiple-input | MIONet [32] | Expanding the network architecture of DeepONet to handle multi-input problems. | Expanding DeepONet to address multi-input problems. | Lack of theoretical support |
| | Fourier-MIONet [112] | Combining DeepONet with U-FNO allows for a significant reduction in computational costs. | Significantly reducing computational costs without compromising prediction accuracy. | Research on different types of equation problems. |
| | MultiAuto-DeepONet [114] | Building a multi-resolution DeepONet to handle inputs from different domains and facilitate solving SDEs. | Handling multi-resolution inputs and reduces the number of trainable parameters. | Lack of research on dynamic nonlinear problems. |
| General | GNOT [31] | Developing a new general neural operator architecture based on transformers. | Simultaneously addressing complex problems, and achieving significant improvements. | Primarily applying to large-scale datasets. |

## References


[1] E.C. Zachmanoglou, D.W. Thoe, Introduction to Partial Differential Equations with Applications, Courier Corporation, 1986.
[2] D.S. Jones, M. Plank, B.D. Sleeman, Differential Equations and Mathematical Biology, CRC Press, 2009.
[3] A. Sommerfeld, Partial Differential Equations in Physics, Academic Press, 1949.
[4] S.K. Godunov, I. Bohachevsky, Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics, Matematičeskij Sb. 47 (3) (1959) 271–306.
[5] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, The Finite Element Method: Its Basis and Fundamentals, Elsevier, 2005.
[6] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (4) (1995) 911–917.
[7] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[8] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: learning PDEs from data: international conference on machine learning, Stock. Schwed. (2018).
[9] S. Karumuri, R. Tripathy, I. Bilionis, J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks, J. Comput. Phys. 404 (2020) 109120.
[10] K. Wu, D. Xiu, Data-driven deep learning of partial differential equations in modal space, J. Comput. Phys. 408 (2020) 109307.
[11] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614.
[12] X. Zhang, T. Cheng, L. Ju, Implicit form neural network for learning scalar hyperbolic conservation laws, in: Mathematical and Scientific Machine Learning, PMLR, 2022, pp. 1082–1098.
[13] D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, S. Hoyer, Machine learning–accelerated computational fluid dynamics, Proc. Natl. Acad. Sci. 118 (21) (2021) e2101784118.
[14] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.
[15] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[16] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[17] T. Tripura, S. Chakraborty, Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems, Comput. Methods Appl. Mech. Engrg. 404 (2023) 115783.
[18] Z. Zhang, L. Wing Tat, H. Schaeffer, Belnet: Basis enhanced learning, a mesh-free neural operator, Proc. R. Soc. A 479 (2276) (2023) 20230043.
[19] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2010.08895.
[20] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Graph kernel network for partial differential equations, 2020, arXiv preprint arXiv:2003.03485.
[21] S. Cao, Choose a transformer: Fourier or galerkin, Adv. Neural Inf. Process. Syst. 34 (2021) 24924–24940.
[22] H. Zheng, W. Nie, A. Vahdat, K. Azizzadenesheli, A. Anandkumar, Fast sampling of diffusion models via operator learning, in: International Conference on Machine Learning, PMLR, 2023, pp. 42390–42402.
[23] X. Liao, A. Qin, J. Seidman, J. Wang, W. Wang, P. Perdikaris, Score neural operator: A generative model for learning and generalizing across multiple probability distributions, 2024, arXiv preprint arXiv:2410.08549.
[24] M. Wei, X. Zhang, Super-resolution neural operator, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 18247–18256.
[25] L. Han, X. Zhang, Scalable super-resolution neural operator, in: Proceedings of the 32nd ACM International Conference on Multimedia, 2024, pp. 10036–10045.
[26] W. Cho, S. Cho, H. Jin, J. Jeon, K. Lee, S. Hong, D. Lee, J. Choi, N. Park, Operator-learning-inspired modeling of neural ordinary differential equations, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 11543–11551.
[27] S. Pal, H. Adepu, C. Wang, P. Golland, V. Singh, Implicit representations via operator learning, in: Forty-First International Conference on Machine Learning, 2024.
[28] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Learning maps between function spaces with applications to PDEs, J. Mach. Learn. Res. 24 (89) (2023) 1–97.

[29] S. Goswami, A. Bora, Y. Yu, G.E. Karniadakis, Physics-informed deep neural operator networks, in: Machine Learning in Modeling and Simulation: Methods and Applications, Springer, 2023, pp. 219–254.

[30] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.

[31] Z. Hao, Z. Wang, H. Su, C. Ying, Y. Dong, S. Liu, Z. Cheng, J. Song, J. Zhu, Gnot: A general neural operator transformer for operator learning, in: International Conference on Machine Learning, PMLR, 2023, pp. 12556–12569.

[32] P. Jin, S. Meng, L. Lu, MIONet: Learning multiple-input operators via tensor product, SIAM J. Sci. Comput. 44 (6) (2022) A3490–A3514.

[33] S. Cai, Z. Wang, L. Lu, T.A. Zaki, G.E. Karniadakis, DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks, J. Comput. Phys. 436 (2021) 110296.

[34] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013, arXiv preprint arXiv:1312.6203.

[35] E.J. Nyström, Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben, Acta Math. 54 (1930) 185–204, URL https://api.semanticscholar.org/CorpusID:122921569.

[36] F. Alet, A.K. Jeewajee, M.B. Villalonga, A. Rodriguez, T. Lozano-Perez, L. Kaelbling, Graph element networks: adaptive, structured computation and memory, in: International Conference on Machine Learning, PMLR, 2019, pp. 212–222.

[37] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, A. Anandkumar, Multipole graph neural operator for parametric partial differential equations, Adv. Neural Inf. Process. Syst. 33 (2020) 6755–6766.

[38] M. Mathieu, M. Henaff, Y. LeCun, Fast training of convolutional networks through ffts, 2013, arXiv preprint arXiv:1312.5851.

[39] Z.J. Xu, Understanding training and generalization in deep learning by fourier analysis, 2018, arXiv preprint arXiv:1808.04295.

[40] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, J. Zhu, Physics-informed machine learning: A survey on problems, methods and applications, 2022, arXiv preprint arXiv:2211.08064.

[41] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, 2015, pp. 234–241.

[42] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, et al., Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, 2022, arXiv preprint arXiv:2202.11214.

[43] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, arXiv preprint arXiv:1409.0473.

[44] H. You, Y. Yu, M. D'Elia, T. Gao, S. Silling, Nonlocal kernel network (nkn): a stable and resolution-independent deep neural network, J. Comput. Phys. 469 (2022) 111536.

[45] I. Daubechies, Ten Lectures on Wavelets, SIAM, 1992.

[46] G. Gupta, X. Xiao, P. Bogdan, Multiwavelet-based operator learning for differential equations, Adv. Neural Inf. Process. Syst. 34 (2021) 24048–24062.

[47] Q. Cao, S. Goswami, T. Tripura, S. Chakraborty, G.E. Karniadakis, Deep neural operators can predict the real-time response of floating offshore structures under irregular waves, Comput. Struct. 291 (2024) 107228.

[48] Q. Cao, S. Goswami, G.E. Karniadakis, Laplace neural operator for solving differential equations, Nat. Mach. Intell. 6 (6) (2024) 631–640.

[49] G. Chen, X. Liu, Y. Li, Q. Meng, L. Chen, Laplace neural operator for complex geometries, 2023, arXiv preprint arXiv:2302.08166.

[50] J.P. Boyd, Chebyshev and Fourier Spectral Methods, Courier Corporation, 2001.

[51] V. Fanaskov, I. Oseledets, Spectral neural operators, 2022, arXiv preprint arXiv:2205.10573.

[52] M.V. de Hoop, D.Z. Huang, E. Qian, A.M. Stuart, The cost-accuracy trade-off in operator learning with neural networks, 2022, arXiv preprint arXiv:2203.13181.

[53] Z. Liu, H. Wang, H. Zhang, K. Bao, X. Qian, S. Song, Render unto numerics: Orthogonal polynomial neural operator for PDEs with nonperiodic boundary conditions, SIAM J. Sci. Comput. 46 (4) (2024) C323–C348, http://dx.doi.org/10.1137/23M1556320, arXiv:https://doi.org/10.1137/23M1556320.

[54] J. Choi, T. Yun, N. Kim, Y. Hong, Spectral operator learning for parametric PDEs without data reliance, Comput. Methods Appl. Mech. Engrg. 420 (2024) 116678, http://dx.doi.org/10.1016/j.cma.2023.116678, URL https://www.sciencedirect.com/science/article/pii/S0045782523008010.

[55] F. Andreu, J. Mazón, J. Rossi, J. Toledo, Nonlocal diffusion problems, volume 165 of mathematical surveys and monographs, Am. Math. Soc. (2010).

[56] G. Gilboa, S. Osher, Nonlocal operators with applications to image processing, Multiscale Model. Simul. 7 (3) (2009) 1005–1028.

[57] X. Wang, R. Girshick, A. Gupta, K. He, Non-local neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7794–7803.

[58] F. Lu, Q. An, Y. Yu, Nonparametric learning of kernels in nonlocal operators, J. Peridynamics Nonlocal Model. (2023) 1–24.

[59] N. Kovachki, S. Lanthaler, S. Mishra, On universal approximation and error bounds for Fourier neural operators, J. Mach. Learn. Res. 22 (1) (2021) 13237–13312.

[60] S. Lanthaler, Z. Li, A.M. Stuart, The nonlocal neural operator: Universal approximation, 2023, arXiv preprint arXiv:2304.13221.

[61] A. Tran, A. Mathews, L. Xie, C.S. Ong, Factorized fourier neural operators, 2021, arXiv preprint arXiv:2111.13802.

[62] E. Popov, M. Neviere, Maxwell equations in Fourier space: fast-converging formulation for diffraction by arbitrary shaped, periodic, anisotropic media, J. Opt. Soc. Amer. A 18 (11) (2001) 2886–2894.

[63] K. Bhattacharya, B. Hosseini, N.B. Kovachki, A.M. Stuart, Model reduction and neural networks for parametric PDEs, SMAI J. Comput. Math. 7 (2021) 121–157.

[64] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, G.E. Karniadakis, A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data, Comput. Methods Appl. Mech. Engrg. 393 (2022) 114778.

[65] M. Rafiq, G. Rafiq, H.-Y. Jung, G.S. Choi, SSNO: Spatio-spectral neural operator for functional space learning of partial differential equations, IEEE Access 10 (2022) 15084–15095.

[66] W. Xiong, X. Huang, Z. Zhang, R. Deng, P. Sun, Y. Tian, Koopman neural operator as a mesh-free solver of non-linear partial differential equations, J. Comput. Phys. (2024) 113194.

[67] S. Wang, H. Wang, P. Perdikaris, Improved architectures and training algorithms for deep operator networks, J. Sci. Comput. 92 (2) (2022) 35.

[68] G.V. Trunk, A problem of dimensionality: A simple example, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1 (3) (1979) 306–307, http://dx.doi.org/10.1109/TPAMI.1979.4766926.

[69] H. Wu, T. Hu, H. Luo, J. Wang, M. Long, Solving high-dimensional PDEs with latent spectral models, 2023, arXiv preprint arXiv:2301.12664.

[70] K. Kontolati, S. Goswami, G.E. Karniadakis, M.D. Shields, Learning in latent spaces improves the predictive accuracy of deep neural operators, 2023, arXiv preprint arXiv:2304.07599.

[71] V. Oommen, K. Shukla, S. Goswami, R. Dingreville, G.E. Karniadakis, Learning two-phase microstructure evolution using neural operators and autoencoder architectures, Npj Comput. Mater. 8 (1) (2022) 190.

[72] R.L. Nowack, Wavefronts and solutions of the eikonal equation, Geophys. J. Int. 110 (1) (1992) 55–62.

[73] E. Haghighat, U. bin Waheed, G. Karniadakis, En-DeepONet: An enrichment approach for enhancing the expressivity of neural operators with applications to seismology, Comput. Methods Appl. Mech. Engrg. 420 (2024) 116681.

[74] J. Seidman, G. Kissas, P. Perdikaris, G.J. Pappas, NOMAD: Nonlinear manifold decoders for operator learning, Adv. Neural Inf. Process. Syst. 35 (2022) 5601–5613.

[75] N. Navaneeth, S. Chakraborty, Waveformer for modeling dynamical systems, Mech. Syst. Signal Process. 211 (2024) 111253.

[76] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, Sci. Adv. 7 (40) (2021) eabi8605.

[77] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, 2021, arXiv preprint arXiv:2111.03794.

[78] N. Navaneeth, T. Tripura, S. Chakraborty, Physics informed WNO, Comput. Methods Appl. Mech. Engrg. 418 (2024) 116546.

[79] S. Wang, M.A. Bhouri, P. Perdikaris, Fast PDE-constrained optimization via self-supervised operator learning, 2021, arXiv preprint arXiv:2110.13297.

[80] S. Goswami, M. Yin, Y. Yu, G.E. Karniadakis, A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials, Comput. Methods Appl. Mech. Engrg. 391 (2022) 114587.

[81] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Marchine Learn. Res. 18 (2018) 1–43.

[82] S. Wu, A. Zhu, Y. Tang, B. Lu, Solving parametric elliptic interface problems via interfaced operator network, J. Comput. Phys. 514 (2024) 113217, http://dx.doi.org/10.1016/j.jcp.2024.113217, URL https://www.sciencedirect.com/science/article/pii/S0021999124004662.

[83] H. Maust, Z. Li, Y. Wang, D. Leibovici, O. Bruno, T. Hou, A. Anandkumar, Fourier continuation for exact derivative computation in physics-informed neural operators, 2022, arXiv preprint arXiv:2211.15960.

[84] M.G. Fernández-Godino, C. Park, N.-H. Kim, R.T. Haftka, Review of multi-fidelity models, 2016, arXiv preprint arXiv:1609.07196.

[85] L. Lu, R. Pestourie, S.G. Johnson, G. Romano, Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport, Phys. Rev. Res. 4 (2) (2022) 023210.

[86] A.A. Howard, M. Perego, G.E. Karniadakis, P. Stinis, Multifidelity deep operator networks for data-driven and physics-informed problems, J. Comput. Phys. 493 (2023) 112462.

[87] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems, J. Comput. Phys. 401 (2020) 109020.

[88] Z. Zhang, F. Bao, L. Ju, G. Zhang, TransNet: Transferable neural networks for partial differential equations, 2023, arXiv preprint arXiv:2301.11701.

[89] S. Goswami, K. Kontolati, M.D. Shields, G.E. Karniadakis, Deep transfer operator learning for partial differential equations under conditional shift, Nat. Mach. Intell. 4 (12) (2022) 1155–1164.

[90] Z. Mao, L. Lu, O. Marxen, T.A. Zaki, G.E. Karniadakis, DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators, J. Comput. Phys. 447 (2021) 110698.

[91] E. Barnard, L. Wessels, Extrapolation and interpolation in neural network classifiers, IEEE Control Syst. Mag. 12 (5) (1992) 50–53.

[92] M. Zhu, H. Zhang, A. Jiao, G.E. Karniadakis, L. Lu, Reliable extrapolation of deep neural operators informed by physics or sparse observations, Comput. Methods Appl. Mech. Engrg. 412 (2023) 116064.

[93] H. You, Q. Zhang, C.J. Ross, C.-H. Lee, M.-C. Hsu, Y. Yu, A physics-guided neural operator learning approach to model biological tissues from digital image correlation measurements, J. Biomech. Eng. 144 (12) (2022) 121012.

[94] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548–26560.

[95] T. De Ryck, F. Bonnet, S. Mishra, E. de Bézenac, An operator preconditioning perspective on training in physics-informed machine learning, 2023, arXiv preprint arXiv:2310.05801.

[96] J. Fan, J. Meng, J. Ludescher, X. Chen, Y. Ashkenazy, J. Kurths, S. Havlin, H.J. Schellnhuber, Statistical physics approaches to the complex earth system, Phys. Rep. 896 (2021) 1–84.

[97] W. Huang, R.D. Russell, Adaptive Moving Mesh Methods, vol. 174, Springer Science & Business Media, 2010.

[98] Z. Li, D.Z. Huang, B. Liu, A. Anandkumar, Fourier neural operator with learned deformations for pdes on general geometries, 2022, arXiv preprint arXiv:2207.05209.

[99] Z. Li, K. Meidani, A.B. Farimani, Transformer for partial differential equations' operator learning, 2022, arXiv preprint arXiv:2205.13671.

[100] E. Weinan, Principles of Multiscale Modeling, Cambridge University Press, 2011.

[101] L. Liu, W. Cai, Multiscale DeepONet for nonlinear operators in oscillatory function spaces for building seismic wave responses, 2021, arXiv preprint arXiv:2111.04860.

[102] M. Yin, E. Zhang, Y. Yu, G.E. Karniadakis, Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems, Comput. Methods Appl. Mech. Engrg. 402 (2022) 115027.

[103] G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, S.M. Benson, U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow, Adv. Water Resour. 163 (2022) 104180.

[104] H. You, Q. Zhang, C.J. Ross, C.-H. Lee, Y. Yu, Learning deep implicit Fourier neural operators (IFNOs) with applications to heterogeneous material modeling, Comput. Methods Appl. Mech. Engrg. 398 (2022) 115296.

[105] B. Lütjens, C.H. Crawford, C.D. Watson, C. Hill, D. Newman, Multiscale neural operator: Learning fast and grid-independent pde solvers, 2022, arXiv preprint arXiv:2207.11417.

[106] X. Liu, B. Xu, L. Zhang, Ht-net: Hierarchical transformer based operator learning model for multiscale pdes, 2022, arXiv preprint arXiv:2210.10890.

[107] X. Liu, B. Xu, S. Cao, L. Zhang, Mitigating spectral bias for the multiscale operator learning, J. Comput. Phys. 506 (2024) 112944.

[108] P.C. Di Leoni, L. Lu, C. Meneveau, G. Karniadakis, T.A. Zaki, Deeponet prediction of linear instability waves in high-speed boundary layers, 2021, arXiv preprint arXiv:2105.08697.

[109] T. Tripura, A. Awasthi, S. Roy, S. Chakraborty, A wavelet neural operator based elastography for localization and quantification of tumors, Comput. Methods Programs Biomed. 232 (2023) 107436.

[110] J. Gu, Z. Gao, C. Feng, H. Zhu, R. Chen, D. Boning, D. Pan, NeurO-Light: A physics-agnostic neural operator enabling parametric photonic device simulation, Adv. Neural Inf. Process. Syst. 35 (2022) 14623–14636.

[111] C. Lin, Z. Li, L. Lu, S. Cai, M. Maxey, G.E. Karniadakis, Operator learning for predicting multiscale bubble growth dynamics, J. Chem. Phys. 154 (10) (2021).

[112] Z. Jiang, M. Zhu, D. Li, Q. Li, Y.O. Yuan, L. Lu, Fourier-MIONet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration, 2023, arXiv preprint arXiv:2303.04778.

[113] Z. Zhang, C. Moya, L. Lu, G. Lin, H. Schaeffer, D2no: Efficient handling of heterogeneous input function spaces with distributed deep neural operators, Comput. Methods Appl. Mech. Engrg. 428 (2024) 117084.

[114] J. Zhang, S. Zhang, G. Lin, Multiauto-deeponet: A multi-resolution autoencoder deeponet for nonlinear dimension reduction, uncertainty quantification and operator learning of forward and inverse stochastic problems, 2022, arXiv preprint arXiv:2204.03193.

[115] G. Lin, C. Moya, Z. Zhang, B-DeepONet: An enhanced Bayesian DeepONet for solving noisy parametric PDEs using accelerated replica exchange SGLD, J. Comput. Phys. 473 (2023) 111713.

[116] S. Garg, S. Chakraborty, VB-DeepONet: A Bayesian operator learning framework for uncertainty quantification, Eng. Appl. Artif. Intell. 118 (2023) 105685.

[117] A. Pensoneault, X. Zhu, Uncertainty quantification for DeepONets with ensemble Kalman inversion, J. Comput. Phys. 523 (2025) 113670.

[118] E. Magnani, M. Pförtner, T. Weber, P. Hennig, Linearization turns neural operators into function-valued Gaussian processes, 2024, arXiv preprint arXiv:2406.05072.

[119] S. Kumar, R. Nayek, S. Chakraborty, Neural operator induced Gaussian process framework for probabilistic solution of parametric partial differential equations, Comput. Methods Appl. Mech. Engrg. 431 (2024) 117265.

[120] S. Kumar, R. Nayek, S. Chakraborty, Towards Gaussian process for operator learning: An uncertainty aware resolution independent operator learning algorithm for computational mechanics, Comput. Methods Appl. Mech. Engrg. 435 (2025) 117664.

[121] C. Mora, A. Yousefpour, S. Hosseinmardi, H. Owhadi, R. Bostanabad, Operator learning with Gaussian processes, Comput. Methods Appl. Mech. Engrg. 434 (2025) 117581.

[122] C. Bülte, P. Scholl, G. Kutyniok, Probabilistic neural operators for functional uncertainty quantification, 2025, arXiv preprint arXiv:2502.12902.

[123] K. Haitsiukevich, O. Poyraz, P. Marttinen, A. Ilin, Diffusion models as probabilistic neural operators for recovering unobserved states of dynamical systems, in: 2024 IEEE 34th International Workshop on Machine Learning for Signal Processing, MLSP, IEEE, 2024, pp. 1–6.

[124] B. Bonev, T. Kurth, C. Hundt, J. Pathak, M. Baust, K. Kashinath, A. Anandkumar, Spherical fourier neural operators: Learning stable dynamics on the sphere, in: International Conference on Machine Learning, PMLR, 2023, pp. 2806–2823.

[125] K. Lin, X. Li, Y. Ye, S. Feng, B. Zhang, G. Xu, Z. Wang, Spherical neural operator network for global weather prediction, IEEE Trans. Circuits Syst. Video Technol. 34 (6) (2023) 4899–4913.

[126] Y. Hu, F. Yin, W. Zhang, K. Ren, J. Song, K. Deng, D. Zhang, Spherical multigrid neural operator for improving autoregressive global weather forecasting, Sci. Rep. 15 (1) (2025) 11522.

[127] G. Kissas, J.H. Seidman, L.F. Guilhoto, V.M. Preciado, G.J. Pappas, P. Perdikaris, Learning operators with coupled attention, J. Mach. Learn. Res. 23 (215) (2022) 1–63.

[128] T. Kurth, S. Subramanian, P. Harrington, J. Pathak, M. Mardani, D. Hall, A. Miele, K. Kashinath, A. Anandkumar, Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators, in: Proceedings of the Platform for Advanced Scientific Computing Conference, 2023, pp. 1–11.

[129] J. Leinonen, B. Bonev, T. Kurth, Y. Cohen, Modulated adaptive Fourier neural operators for temporal interpolation of weather forecasts, 2024, arXiv preprint arXiv:2410.18904.

[130] H. Zhang, Y. Wang, Y. Jian, J. Jiang, Z. Bai, L. Ma, Climate downscaling using neural operator: Spatiotemporal multimodal fusion operator with state-query coupled kernel, in: ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2025, pp. 1–5.

[131] P. Jiang, Z. Yang, J. Wang, C. Huang, P. Xue, T. Chakraborty, X. Chen, Y. Qian, Efficient super-resolution of near-surface climate modeling using the Fourier neural operator, J. Adv. Model. Earth Syst. 15 (7) (2023) e2023MS003800.

[132] Q. Yang, A. Hernandez-Garcia, P. Harder, V. Ramesh, P. Sattigeri, D. Szwarcman, C.D. Watson, D. Rolnick, Fourier neural operators for arbitrary resolution climate data downscaling, J. Mach. Learn. Res. 25 (420) (2024) 1–30.

[133] A. Bora, K. Shukla, S. Zhang, B. Harrop, R. Leung, G.E. Karniadakis, Learning bias corrections for climate models using deep neural operators, 2023, arXiv preprint arXiv:2302.03173.

[134] S. Bedi, K. Tiwari, P. AP, S.H. Kota, N.A. Krishnan, A neural operator for forecasting carbon monoxide evolution in cities, Npj Clean Air 1 (1) (2025) 2.

[135] G. Wen, Z. Li, Q. Long, K. Azizzadenesheli, A. Anandkumar, S.M. Benson, Real-time high-resolution CO 2 geological storage prediction using nested Fourier neural operators, Energy Environ. Sci. 16 (4) (2023) 1732–1741.

[136] T. Kadeethum, S.J. Verzi, H. Yoon, An improved neural operator framework for large-scale CO2 storage operations, Geoenergy Sci. Eng. 240 (2024) 213007.

[137] H. Tang, Q. Kong, J.P. Morris, Multi-fidelity Fourier neural operator for fast modeling of large-scale geological carbon storage, J. Hydrol. 629 (2024) 130641.

[138] K. Qi, J. Sun, Gabor-filtered fourier neural operator for solving partial differential equations, Comput. & Fluids 274 (2024) 106239.

[139] A. Chandra, M. Koch, S. Pawar, A. Panda, K. Azizzadenesheli, J. Snippe, F.O. Alpak, F. Hariri, C. Etienam, P. Devarakota, et al., Fourier neural operator based surrogates for $CO\_2$ storage in realistic geologies, 2025, arXiv preprint arXiv:2503.11031.

[140] P. Rivera-Casillas, S. Dutta, S. Cai, M. Loveland, K. Nath, K. Shukla, C. Trahan, J. Lee, M. Farthing, C. Dawson, A neural operator-based emulator for regional shallow water dynamics, 2025, arXiv preprint arXiv:2502.14782.

[141] B. Chen, Z. Sheng, F. Cui, Refined short-term forecasting atmospheric temperature profiles in the stratosphere based on operators learning of neural networks, Earth Space Sci. 11 (4) (2024) e2024EA003509.

[142] H. Wu, K. Weng, S. Zhou, X. Huang, W. Xiong, Neural manifold operators for learning the evolution of physical dynamics, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 3356–3366.

[143] X. Wang, P. Li, K. Jia, S. Zhang, C. Li, B. Wu, Y. Dong, D. Lu, SPI-MIONet for surrogate modeling in phase-field hydraulic fracturing, Comput. Methods Appl. Mech. Engrg. 427 (2024) 117054.
[144] P.C. Di Leoni, L. Lu, C. Meneveau, G.E. Karniadakis, T.A. Zaki, Neural operator prediction of linear instability waves in high-speed boundary layers, J. Comput. Phys. 474 (2023) 111793.
[145] C. Li, H. Zhao, Y. Hao, A feature enhanced autoencoder integrated with Fourier neural operator for intelligent elastic wavefield modeling, IEEE Trans. Geosci. Remote Sens. (2025).
[146] F. Lehmann, F. Gatti, D. Clouteau, Multiple-input fourier neural operator (mifno) for source-dependent 3d elastodynamics, J. Comput. Phys. (2025) 113813.
[147] F. Lehmann, F. Gatti, M. Bertin, D. Clouteau, 3D elastic wave propagation with a factorized Fourier neural operator (F-FNO), Comput. Methods Appl. Mech. Engrg. 420 (2024) 116718.
[148] R. Li, W. Ye, Y. Liu, A boundary-based fourier neural operator (B-FNO) method for efficient parametric acoustic wave analysis, Eng. Comput. (2025) 1–18.
[149] A. Abdullin, U.B. Waheed, K. Suleymanli, F. Stanek, Microseismic source localization using Fourier neural operator with application to field data from utah FORGE, IEEE Trans. Geosci. Remote Sens. (2025).
[150] Y. Mei, Y. Zhang, X. Zhu, R. Gou, J. Gao, Fully convolutional network enhanced DeepONet-based surrogate of predicting the travel-time fields, IEEE Trans. Geosci. Remote Sens. (2024).
[151] N. Borrel-Jensen, S. Goswami, A.P. Engsig-Karup, G.E. Karniadakis, C.-H. Jeong, Sound propagation in realistic interactive 3D scenes with parameterized sources using deep neural operators, Proc. Natl. Acad. Sci. 121 (2) (2024) e2312159120.
[152] J. He, S. Koric, S. Kushwaha, J. Park, D. Abueidda, I. Jasiuk, Novel DeepONet architecture to predict stresses in elastoplastic structures with variable complex geometries and loads, Comput. Methods Appl. Mech. Engrg. 415 (2023) 116277.
[153] S. Kushwaha, J. Park, S. Koric, J. He, I. Jasiuk, D. Abueidda, Advanced deep operator networks to predict multiphysics solution fields in materials processing and additive manufacturing, Addit. Manuf. 88 (2024) 104266.
[154] S. Jafarzadeh, S. Silling, N. Liu, Z. Zhang, Y. Yu, Peridynamic neural operators: A data-driven nonlocal constitutive model for complex material responses, Comput. Methods Appl. Mech. Engrg. 425 (2024) 116914.
[155] S. Rezaei, R.N. Asl, S. Faroughi, M. Asgharzadeh, A. Harandi, R.N. Koopas, G. Laschet, S. Reese, M. Apel, A finite operator learning technique for mapping the elastic properties of microstructures to their mechanical deformations, Internat. J. Numer. Methods Engrg. 126 (1) (2025) e7637.
[156] C. Kelly, S.R. Kalidindi, Thermodynamically-informed iterative neural operators for heterogeneous elastic localization, Comput. Methods Appl. Mech. Engrg. 441 (2025) 117939.
[157] A. Kashefi, T. Mukerji, A novel Fourier neural operator framework for classification of multi-sized images: Application to three dimensional digital porous media, Phys. Fluids 36 (5) (2024).
[158] W. Peng, Z. Yuan, J. Wang, Attention-enhanced neural network models for turbulence simulation, Phys. Fluids 34 (2) (2022).
[159] X. Ye, H. Li, J. Huang, G. Qin, On the locality of local neural operator in learning fluid dynamics, Comput. Methods Appl. Mech. Engrg. 427 (2024) 117035.
[160] K. Shukla, V. Oommen, A. Peyvan, M. Penwarden, N. Plewacki, L. Bravo, A. Ghoshal, R.M. Kirby, G.E. Karniadakis, Deep neural operators as accurate surrogates for shape optimization, Eng. Appl. Artif. Intell. 129 (2024) 107615.
[161] J. He, S. Koric, D. Abueidda, A. Najafi, I. Jasiuk, Geom-deeponet: A point-cloud-based deep operator network for field predictions on 3d parameterized geometries, Comput. Methods Appl. Mech. Engrg. 429 (2024) 117130.
[162] Y. Xie, B. Deng, C. Jiang, C. Lv, A boundary-assimilation Fourier neural operator for predicting initial fields of flow around structures, Phys. Fluids 37 (2) (2025).
[163] M. Ramezankhani, A. Deodhar, R.Y. Parekh, D. Birru, An advanced physics-informed neural operator for comprehensive design optimization of highly-nonlinear systems: An aerospace composites processing case study, Eng. Appl. Artif. Intell. 142 (2025) 109886.
[164] S. Goswami, D.S. Li, B.V. Rego, M. Latorre, J.D. Humphrey, G.E. Karniadakis, Neural operator learning of heterogeneous mechanobiological insults contributing to aortic aneurysms, J. R. Soc. Interface 19 (193) (2022) 20220410.
[165] M. Yin, E. Ban, B.V. Rego, E. Zhang, C. Cavinato, J.D. Humphrey, G. Em Karniadakis, Simulating progressive intramural damage leading to aortic dissection using DeepONet: an operator–regression neural network, J. R. Soc. Interface 19 (187) (2022) 20210670.
[166] O.L. Cruz-González, V. Deplano, B. Ghattas, Enhanced vascular flow simulations in aortic aneurysm via physics-informed neural networks and deep operator networks, 2025, arXiv preprint arXiv:2503.17402.
[167] J. Su, J. Ma, S. Tong, E. Xu, M. Chen, Multiscale attention wavelet neural operator for capturing steep trajectories in biochemical systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 15100–15107.
[168] M. Laudato, L. Manzari, K. Shukla, High-fidelity description of platelet deformation using a neural operator, 2024, arXiv preprint arXiv:2412.00747.
[169] M. Laudato, L. Manzari, K. Shukla, Neural operator modeling of platelet geometry and stress in shear flow, 2025, arXiv preprint arXiv:2503.12074.
[170] Z. Zeng, Y. Zheng, Y. Zheng, Y. Li, Z. Shi, H. Sun, Neural Born series operator for biomedical ultrasound computed tomography, 2023, arXiv preprint arXiv:2312.15575.
[171] H. Dai, M. Penwarden, R.M. Kirby, S. Joshi, Neural operator learning for ultrasound tomography inversion, 2023, arXiv preprint arXiv:2304.03297.
[172] T. Zhou, X. Wan, D.Z. Huang, Z. Li, Z. Peng, A. Anandkumar, J.F. Brady, P.W. Sternberg, C. Daraio, AI-aided geometric design of anti-infection catheters, Sci. Adv. 10 (1) (2024) eadj1741.
[173] E. Magnani, N. Krämer, R. Eschenhagen, L. Rosasco, P. Hennig, Approximate Bayesian neural operators: Uncertainty quantification for parametric PDEs, 2022, arXiv preprint arXiv:2208.01565.
[174] L. Guo, H. Wu, Y. Wang, W. Zhou, T. Zhou, IB-UQ: Information bottleneck based uncertainty quantification for neural function regression and neural operator learning, J. Comput. Phys. 510 (2024) 113089.
[175] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081.
[176] P. Rathore, W. Lei, Z. Frangella, L. Lu, M. Udell, Challenges in training PINNs: A loss landscape perspective, 2024, arXiv preprint arXiv:2402.01868.
[177] M. McCabe, B.R.-S. Blancard, L.H. Parker, R. Ohana, M. Cranmer, A. Bietti, M. Eickenberg, S. Golkar, G. Krawezik, F. Lanusse, et al., Multiple physics pretraining for physical surrogate models, 2023, arXiv preprint arXiv:2310.02994.
[178] J. Hu, P. Jin, A hybrid iterative method based on MIONet for PDEs: Theory and numerical examples, 2024, arXiv preprint arXiv:2402.07156.
[179] O. Ovadia, V. Oommen, A. Kahana, A. Peyvan, E. Turkel, G.E. Karniadakis, Real-time inference and extrapolation via a diffusion-inspired temporal transformer operator (DiTTO), 2023, arXiv preprint arXiv:2307.09072v2.
[180] O. Ovadia, A. Kahana, P. Stinis, E. Turkel, D. Givoli, G.E. Karniadakis, Vito: Vision transformer-operator, Comput. Methods Appl. Mech. Engrg. 428 (2024) 117109.
[181] Z. Hu, N.A. Daryakenari, Q. Shen, K. Kawaguchi, G.E. Karniadakis, State-space models are accurate and efficient neural operators for dynamical systems, 2024, arXiv preprint arXiv:2409.03231.
[182] Z. Hao, J. Yao, C. Su, H. Su, Z. Wang, F. Lu, Z. Xia, Y. Zhang, S. Liu, L. Lu, et al., Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes, 2023, arXiv preprint arXiv:2306.08827.
[183] A. Yu, C. Becquey, D. Halikias, M.E. Mallory, A. Townsend, Arbitrary-depth universal approximation theorems for operator neural networks, 2021, arXiv preprint arXiv:2109.11354.
[184] B. Deng, Y. Shin, L. Lu, Z. Zhang, G.E. Karniadakis, Convergence rate of DeepONets for learning operators arising from advection-diffusion equations, 2021, arXiv preprint arXiv:2102.10621.
[185] S. Lanthaler, S. Mishra, G.E. Karniadakis, Error estimates for deeponets: A deep learning framework in infinite dimensions, Trans. Math. Appl. 6 (1) (2022) tnac001.
[186] C. Marcati, C. Schwab, Exponential convergence of deep operator networks for elliptic partial differential equations, SIAM J. Numer. Anal. 61 (3) (2023) 1513–1545.
[187] T. De Ryck, S. Mishra, Generic bounds on the approximation error for physics-informed (and) operator learning, Adv. Neural Inf. Process. Syst. 35 (2022) 10945–10958.
[188] S. Lanthaler, Operator learning with PCA-Net: upper and lower complexity bounds, 2023, arXiv preprint arXiv:2303.16317.
[189] N.B. Kovachki, S. Lanthaler, A.M. Stuart, Operator learning: Algorithms and analysis, 2024, arXiv preprint arXiv:2402.15715.

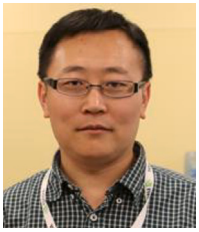
**Shengjun Liu** is a full professor in the School of Mathematics and Statistics at Central South University, China. His research interests include geometric modeling, reverse engineering, computer graphics, and computer-aided geometric design. Liu has a DEng in computer science and technology from Zhejiang University. Contact him at shjliu.cg@csu.edu.cn.

**Yu Yu** is a third-year master's student in the School of Mathematics and Statistics, Central South University, China, supervised by professor Shengjun Liu and Xinru Liu. Her research interests include machine learning and scientific computing, neural operator for solving PDEs. Contact her at yuyu2022@csu.edu.cn.

**Ting Zhang** is a fourth-year Ph.D. student in the School of Mathematics and Statistics, Central South University, China, supervised by professor Shengjun Liu. Her research interests include machine learning and scientific computing, computer vision, and computer-aided geometric design. Contact her at tingzh_csu@foxmail.com.

**Hanchao Liu** is a third-year Ph.D. student in the School of Mathematics and Statistics, Central South University, China, supervised by professor Shengjun Liu. His research interests include machine learning and scientific computing, neural operator for solving PDEs, computer vision, and computer-aided geometric design. Contact him at hchliu@csu.edu.cn.

**Xinru Liu** is an associate professor in the School of Mathematics and Statistics at Central South University, China. He received Ph.D. in applied mathematics and now is working on computer-aided geometric design, numerical optimization, and data mining. Contact him at liuxinru@csu.edu.cn.



**Deyu Meng** is a full professor in the Research Institute for Mathematics and Mathematical Technology, Xi'an Jiaotong University, China. His research interests include Machine Learning, Applied Mathematics, Computer Vision, Artificial Intelligence. Contact him at dymeng@mail.xjtu.edu.cn.