Information Fusion 123 (2025) 103255

Contents lists available at ScienceDirect

Information Fusion

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

Full length article

Machine learning for modelling unstructured grid data in computational physics: A review

Sibo Cheng [a,*], Marc Bocquet [a], Weiping Ding [b,d], Tobias Sebastian Finn [a], Rui Fu [c], Jinlong Fu [e,f], Yike Guo [g], Eleda Johnson [h], Siyi Li [h], Che Liu [h], Eric Newton Moro [i], Jie Pan [j,k], Matthew Piggott [h], Cesar Quilodran [h,l], Prakhar Sharma [m], Kun Wang [h], Dunhui Xiao [c], Xiao Xue [n], Yong Zeng [o], Mingrui Zhang [h], Hao Zhou [p], Kewei Zhu [q], Rossella Arcucci [h]

[a] CEREA, ENPC, EDF R&D, Institut Polytechnique de Paris, Île-de-France, France
[b] School of Artificial Intelligence and Computer Science, Nantong University, Nantong, 226019, Jiangsu, China
[c] School of Mathematical Sciences, Key Laboratory of Intelligent Computing and Applications, Tongji University, Shanghai, 200092, China
[d] Faculty of Data Science, City University of Macau, 999078, Macao Special Administrative Region of China
[e] School of Engineering and Materials Science, Faculty of Science and Engineering, Queen Mary University of London, London, E1 4NS, UK
[f] Zienkiewicz Centre for Modelling, Data and AI, Faculty of Science and Engineering, Swansea University, Swansea, SA1 8EN, UK
[g] Department of Computer Science and Engineering, Hong Kong university of science and technology, Hong Kong, China
[h] Department of Earth Science & Engineering, Imperial College London, London, SW7 2AZ, UK
[i] Tianjin Key Laboratory of Imaging and Sensing Microelectronics Technology, School of Microelectronics, Tianjin University, Tianjin, 300072, China
[j] Centre for Health Informatics, Cumming School of Medicine, University of Calgary, Calgary, T2N 1N4, Alberta (AB), Canada
[k] Department of Community Health Sciences, Cumming School of Medicine, University of Calgary, Calgary, T2N 1N4, Alberta (AB), Canada
[l] Undaunted, Grantham Institute for Climate Change and the Environment, Imperial College London, London, SW7 2AZ, UK
[m] UK Atomic Energy Authority (UKAEA), Culham Campus, Abingdon, OX14 3DB, UK
[n] Centre for Computational Science, Department of Chemistry, University College London, London, WC1H 0AJ, UK
[o] Concordia Institute for Information Systems Engineering, Concordia University, Montreal, H3G 1M8, Quebec (QC), Canada
[p] School of Mechanical, Medical and Process Engineering, Faculty of Engineering, Queensland University of Technology, Brisbane, Queensland (QLD), Australia
[q] Department of Chemical Engineering, University College London, London, WC1E 6BT, UK

ARTICLE INFO

Keywords:
Machine learning
Unstructured data
Reduced order modelling
Adaptive meshes
Computational physics

ABSTRACT

Unstructured grid data are essential for modelling complex geometries and dynamics in computational physics. Yet, their inherent irregularity presents significant challenges for conventional machine learning (ML) techniques. This paper provides a comprehensive review of advanced ML methodologies designed to handle unstructured grid data in high-dimensional dynamical systems. Key approaches discussed include graph neural networks, transformer models with spatial attention mechanisms, interpolation-integrated ML methods, and meshless techniques such as physics-informed neural networks. These methodologies have proven effective across diverse fields, including fluid dynamics and environmental simulations. This review is intended as a guidebook for computational scientists seeking to apply ML approaches to unstructured grid data in their domains, as well as for ML researchers looking to address challenges in computational physics. It places special focus on how ML methods can overcome the inherent limitations of traditional numerical techniques and, conversely, how insights from computational physics can inform ML development. For this purpose, we mainly focus in this review on recent papers from the past decade that reflect strong interactions between computational physics and deep learning methods. To support benchmarking, this review also provides a summary of open-access datasets of unstructured grid data in computational physics. Finally, emerging directions such as generative models with unstructured data, reinforcement learning for mesh generation, and hybrid physics-data-driven paradigms are discussed to inspire future advancements in this evolving field.

Contents

1. Introduction 2
2. Preliminary 3

* Corresponding author.
E-mail address: sibo.cheng@enpc.fr (S. Cheng).

https://doi.org/10.1016/j.inffus.2025.103255
Received 9 February 2025; Received in revised form 18 April 2025; Accepted 20 April 2025
Available online 15 May 2025

2.1. Unstructured meshes in computational physics ... 4
2.2. Reduced order modelling for structured and unstructured data ... 4
2.3. Shallow machine learning for dynamical systems ... 7
2.4. Convolutional and recurrent neural networks ... 7
2.5. Gridding irregular data via interpolation ... 8
3. Machine learning models designed for unstructured grid data ... 10
3.1. Machine learning with preprocessing ... 11
3.1.1. Convolutional neural network with interpolation methods ... 11
3.1.2. Machine learning with mesh reordering and transformation ... 12
3.2. Graph neural networks ... 12
3.3. Transformer and attention mechanism ... 14
3.3.1. Transformers: attention-mechanism ... 14
3.3.2. Applications in computational physics ... 15
3.4. Summary and comparison ... 16
4. Learning paradigms with unstructured data ... 17
4.1. PINNs: a meshless solution ... 17
4.2. Reinforcement learning for unstructured mesh generation ... 19
4.2.1. Introduction to reinforcement learning ... 19
4.2.2. Reinforcement learning for mesh optimisation ... 20
4.3. Generative AI models with unstructured grid data ... 21
4.3.1. Introduction to generative models ... 21
4.3.2. Generative models applied on unstructured data ... 21
5. Public study cases and computational libraries ... 22
5.1. Open-access datasets ... 22
5.2. Open-access computational libraries ... 24
5.2.1. PINN-based PDE solvers and neural operators ... 24
5.2.2. Geometric deep learning and graph-based ML ... 25
6. Discussion and conclusion ... 25
CRediT authorship contribution statement ... 27
Declaration of competing interest ... 27
Acknowledgements ... 27
Data availability ... 27
References ... 27

1. Introduction

Machine learning methods are increasingly being adopted in the field of computational physics to enhance the efficiency of traditional physics-based approaches [1,2]. These methods have shown significant promise in accelerating simulations, reducing computational costs, and improving prediction accuracy. When dealing with high-dimensional dynamical systems, if the data structure is regular both spatially and temporally, mature techniques from image and video processing, such as tree-based models, Multi layer perceptrons (MLPs), CNNs, and Recurrent neural networks (RNNs), can be effectively applied to tasks like field prediction, parameter identification, clustering, and super resolution [3,4]. However, these approaches generally require a square grid structure, homogeneous time steps or fixed input dimensions, which limits their applicability to structured grid data. This becomes a significant challenge in cases where data is irregular or unstructured, as is often encountered in high-fidelity simulations [5,6]. Unstructured data often comes in the form of various mesh or grid types, such as triangles and tetrahedra, quadrilaterals and hexahedra, polyhedral meshes, and adaptively refined meshes. Example application areas include cerebral haemodynamics [7], cardiac electrophysiology [8], environmental modelling [9,10] and DNA rendering [11]. These meshes are essential for accurately representing complex geometries in simulations, but they lack the regular structure that traditional machine learning methods typically require [12].

To address these challenges, there is a growing interest in developing machine learning techniques that can manage unstructured data, common in many areas of computational physics. An important family of approaches involves using machine learning models in conjunction with grid interpolation (e.g., nearest neighbour [13]) and reordering techniques (e.g., space-filling curves [14]). These methods map unstructured or sparse data onto a structured grid, enabling the application of traditional deep learning models such as CNN. A recent representative work is the Voronoi-tessellation-assisted CNN [15,16]. Although effective, this approach can introduce interpolation errors and may not fully capture the complexities of the original unstructured data.

Another promising direction is the use of GNNs, which are specifically designed to work with data represented as values on graphs rather than in grids [17]. In computational physics, where the data often consists of points connected in an irregular manner (such as nodes in a mesh), GNNs excel by directly modelling the relationships between these points. Importantly, GNNs can also handle adaptive meshes with a time-varying number of nodes or grids, allowing them to dynamically adjust to changes in the mesh structure during simulations [18,19]. This capability makes GNNs highly effective at capturing local interactions and generalising across various and evolving mesh configurations.

Transformer-type neural networks, equipped with spatial attention mechanisms [20,21], have also shown great potential in handling unstructured data. These models can focus on relevant spatial features regardless of the data's irregular structure, allowing for improved learning and generalisation. Like GNNs, transformers can manage adaptive meshes where the number of nodes or grids changes over time, providing flexibility in scenarios where the relationships between data points are complex and non-local [22]. The ability to dynamically adjust focus based on the importance of different regions in the data makes transformers particularly useful in applications involving complex physical systems with evolving geometries.

PINNs [12,23,24] offer another innovative solution, particularly by enabling meshless predictions. Unlike traditional methods that require a predefined mesh, PINNs can directly incorporate physical laws into the learning process without relying on a specific grid structure. This meshless approach avoids the challenges associated with unstructured meshes. PINNs can effectively model complex physical phenomena

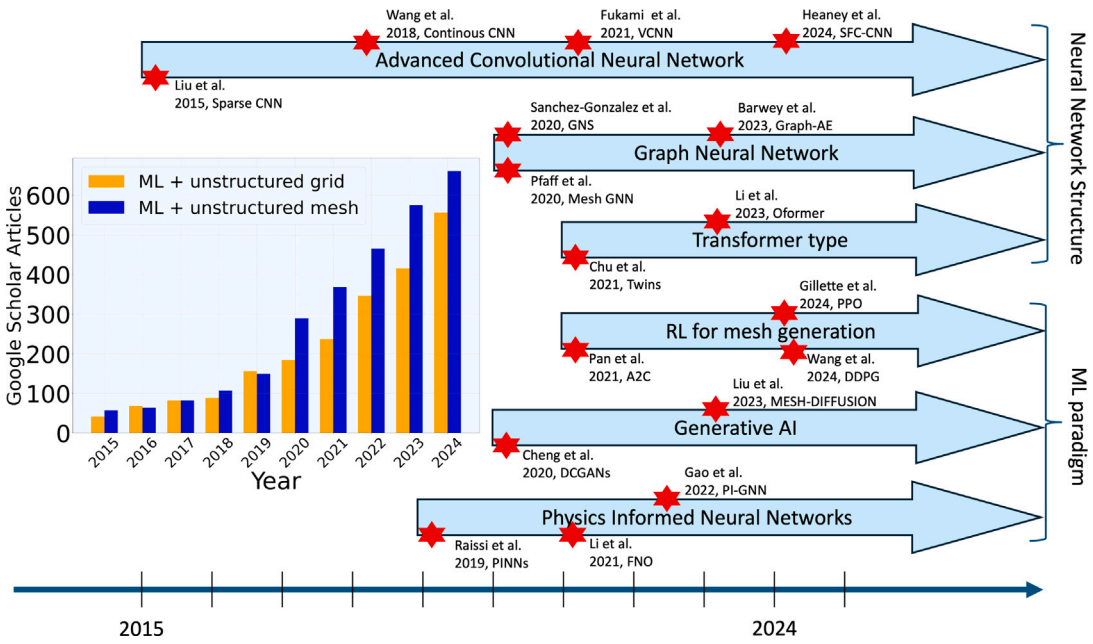


Fig. 1. Timeline of some representative works that apply advanced machine learning techniques for unstructured grid data.

by embedding the governing equations of the system into the neural network architecture [23], ensuring that predictions remain physically consistent. This makes PINNs ideal for a wide range of applications where mesh generation can be particularly challenging [25]. In addition to deep neural networks, efforts have also focused on using shallow ML methods, such as Gaussian processes [26,27] and Random forests [28], to learn pixel-wise mappings from spatiotemporal data points, thereby addressing the challenge of explicitly handling unstructured spatial data. This approach aligns with the meshless PINNs framework.

On the other hand, by adapting the advanced machine learning techniques previously discussed, modern AI paradigms like Reinforcement learning (RL) and generative AI could also play a crucial role in enhancing the modelling of dynamical systems with unstructured grid data. Reinforcement learning is increasingly applied to optimise unstructured meshes by dynamically adjusting mesh elements based on solution variability, reducing reliance on heuristic rules [29]. Its success has been recently observed in simulations of fluid dynamics [30,31]. On the other hand, generating unstructured grid data, particularly for spatial–temporal systems, has been a long-standing challenge in the generative AI community. Efforts have spanned Variational Autoencoder (VAE) [32], Generative adversarial network (GAN) [33], and score-based diffusion models [34,35]. The latter have recently been widely applied to dynamical systems due to their robustness to sparse observations [36,37] and flexibility in conditioning [38,39].

These emerging ML techniques, as shown in Fig. 1, are revolutionising how unstructured data is processed in computational physics, enabling more accurate, efficient, and flexible simulations of complex systems. Fig. 1 also illustrates the number of publications indexed by Google Scholar that contain both 'machine learning' and 'unstructured grid/mesh'. A clear upward trend can be observed, particularly following the surge in deep learning methods around 2018 and 2019. However, there is a lack of a meta-analysis and comparison of these methods in the existing literature, which hinders knowledge transfer and benchmarking across different application domains. This survey explores the latest advancements in machine learning methods specifically tailored for modelling unstructured data, focusing on how these approaches are being adapted to overcome the limitations of traditional techniques and improve the overall efficiency and effectiveness of simulations in complex physical systems. We emphasise that the goal of this work is not to compare the performance of existing methods, as they were designed to solve different challenges. In summary, the key contributions of this paper are as follows:

- To the best of the authors' knowledge, this is the first review paper to comprehensively summarise the development of machine learning techniques for unstructured grid data in dynamical systems within computational science.
- This paper has a special focus on how cutting-edge machine learning techniques, in particular deep neural networks, could tackle the bottleneck of data sparsity and irregularity challenges in simulation data with unstructured or adaptive grids.
- This review covers a variety of significant applications in computational science, such as environmental simulation and multiphase flow modelling involving complex geometries, though it does not provide an in-depth analysis of applications, as the primary focus of this paper is on the methodology.

Given the comprehensive scope of the methods covered in this review and the large body of related research, the literature selection is guided by two principles: milestone studies and citation chaining, with a particular focus on recent research outputs published since 2015. The rest of the paper is organised as follows. Section 2 introduces the concepts of dynamical systems, irregular grid structures, and data-driven techniques in dynamical systems, ranging from reduced-order modelling and shallow machine learning methods to deep neural networks. Section 3 discusses advanced neural network architectures and processing techniques for handling data with unstructured grids. A qualitative comparison is provided at the end of this section. Machine learning paradigms, including PINNs, RL, and generative AI, are introduced in Section 4. We also list current open-access datasets for benchmarking machine learning approaches with unstructured grids in computational physics in Section 5.2.2. The paper concludes with Section 6.

2. Preliminary

In this section, we provide a brief overview of the concept of unstructured grid data in computational science. This is followed by an introduction to ML methods, including both shallow approaches (such

as Random forest (RF) and Gaussian processes) and deep neural networks (primarily CNN and RNN) for dynamical systems. Reduced-order modelling and spatial interpolation techniques are also discussed in this section, as they are important strategies for handling unstructured data and have been integrated into many ML models.

2.1. Unstructured meshes in computational physics

To describe the behaviour or state of dynamical problems, such as airflow around a spacecraft or stress concentration in a dam, numerical simulations require a finite number of points (in time and space) to describe the fields of physical quantities throughout the whole computational domain. Numerical methods, such as the finite difference method, finite volume methods, and finite element analysis, are developed to discretise the equations governing these dynamical problems, allowing for the calculation of system's behaviours over time and space shown by variable values at discrete points throughout the computational domain. In the implementation of these numerical methods, the mesh is the foundation for numerical simulations. Its quality and structure significantly influence the simulations' accuracy, efficiency, and stability, especially in complex simulated domains. To address this situation, unstructured meshes are noted for adaptation on irregular boundaries in simulation-based fluid and solid dynamical systems. Their flexibility and efficiency in handling complex geometries and dynamic problems make them highly suitable for accurately representing the diverse and changing conditions typical of real-world physical systems.

Meshes discretise domains in a space-filling manner such that they provide a description of the associated computational topology and geometry for the fields and equations of functional data. The quality and features of a mesh domain discretisation have been shown through many examples to limit the performance quality of the numerical scheme being employed [40]. Unstructured meshes are advantaged by an arbitrary structure, allowing for flexibility in conformance to geometries and boundaries of the physical system they are used to represent. The topology of meshes typically is arranged hierarchically, built up from nodes connected by edges forming loops which in turn define two-dimensional faces which can be grouped individually or collectively into elements [41]. Geometrically, the node is related to a point, an edge to a curve and a face to a two-dimensional surface. These components, when referenced in a discretised mesh are sometimes referred to as vertex and segment, respectively [42].

In general, unstructured meshes are constructed from space-filling elements of a single or mixed shape-type based on the spatial dimension of the domain [43]. In one dimension, this is a range of finite segments of potentially varying lengths. In two-dimensional surface meshes, a simplex or triangle is the most widely utilised element, though quadrilaterals or a mix of the two are also permissible. Three-dimensional, unstructured volume meshes typically replace the surface mesh triangular base element with a tetrahedral form. Quadrilaterals (pentahedra, hexahedra, prisms, etc.) and other element volumetric shapes are also utilised [44]. The associated two-dimensional surface meshes can serve as a template for constraining the geometry of the space-filling tetrahedra [40].

Unstructured surface and volume meshes are advantaged in their flexibility to conform to potentially complex geometries found in both static and dynamical applications. Controlling the size, shape and/or orientation of the mesh element topology allows for problem-specific optimisation. On one hand, principles of equidistribution [45] are used to generate unstructured isotropic meshes with elements of roughly the same size, shape and orientation. Isotropic meshes are advantageous for capturing turbulent flows and more uniformly distributed phenomena [40]. On the other hand, anisotropic meshes allow for a larger range of element sizes, shapes and orientations. Anisotropic meshes can be constructed or optimised such that the size, shape and distribution of elements are designed to align with similar anisotropic physical features such as those seen in shocks [46] or strongly directional flow such as tidal regimes [47].

The unstructured mesh features which advantage them in complex geometric and multi-scale problem applications can also lead to increased overhead in memory usage, algorithmic complexity, and overall computational cost. For the same node count, unstructured meshes generally have higher memory requirements to store all variations of connectivity and additional topology information unique to arbitrary cells [48]. In applications such as in geometrical volume-of-fluid methods extension, where unstructured meshes allow extension to more complex domains and improved accuracy, implementation has lagged due to the complex geometrical operations required to handle the resultant arbitrary, and non-orthogonal orientations [49]. Often there is a trade-off between conducting fast, stable simulations needed for deployed tool performance and high enough mesh resolution to minimise costly numerical prediction errors [7].

Unstructured meshes are commonly employed in modelling of structures, both to adequately capture complex physical geometries present and to enable more efficient Computational fluid dynamics (CFD) simulations. Quadrilateral meshes (as seen in Fig. 2) are aptly suited to modelling complex, holistic aircraft designs where structural member interactions need to be captured and can accelerate airframe structural analysis in automated workflows [50]. The flexibility of unstructured meshes is further illustrated in the coupling of complex fluid–structure interactions, where evolving flow dynamics along boundary conditions requires compatibility of structure and spatial discretisation between the different domains [51]. The feasibility of a 3D finite-volume method for dynamic fluid–structure interactions was shown in [51] using a loaded fixed-free cantilever wing-like structure in incompressible flow with no turbulence modelling. However, without further optimisation, the cost of extending this workflow to more complex geometries and flow physics, such as flutter dynamics, was predicted to be computationally prohibitive [51].

2.2. Reduced order modelling for structured and unstructured data

Despite current advancements in computer technology, computational mesh sizes remain constrained by available computational resources, especially unstructured mesh, which produces a much larger number of meshes for the same conditions of dynamical problems [5,52]. The Reduced-order modelling (ROM) technique has become a vital method for reducing computational burdens through low-dimensional reduced models [4]. These models are fast to solve and approximate well high-fidelity simulations of dynamic systems [53,54]. Especially, non-intrusive reduced order modellings (NIROMs) have become popular across various research and engineering fields [55–61]. They operate independently as robust models, offering accurate descriptions from high-fidelity simulations without any modifications to original source codes [62]. It is important to note that ROM techniques can also be considered a means of transforming unstructured data into a regular and often fixed-size reduced space, which facilitates downstream tasks such as field prediction or parameter identification.

In ROMs for computational simulations, Proper orthogonal decomposition (POD), also known as Principal component analysis (PCA) or Empirical orthogonal functions (EOF) has proven to be a classic method for spatial dimensionality reduction on data [53]. POD and its variants have been applied with unstructured meshes in various research areas successfully, such as eigenvalue problems in reactor physics [63]; ocean models [64]; aerospace design and optimisation [65]; fluid dynamics with applications in porous media [66], Navier–Stokes Equations [55,67], air pollution [68], and shallow water equations [69]. It is often applied with Galerkin projection to form intrusive ROMs and applied with various interpolation or regression methods, like radial basis function, neural network in NIROMs [56,62,70,71].

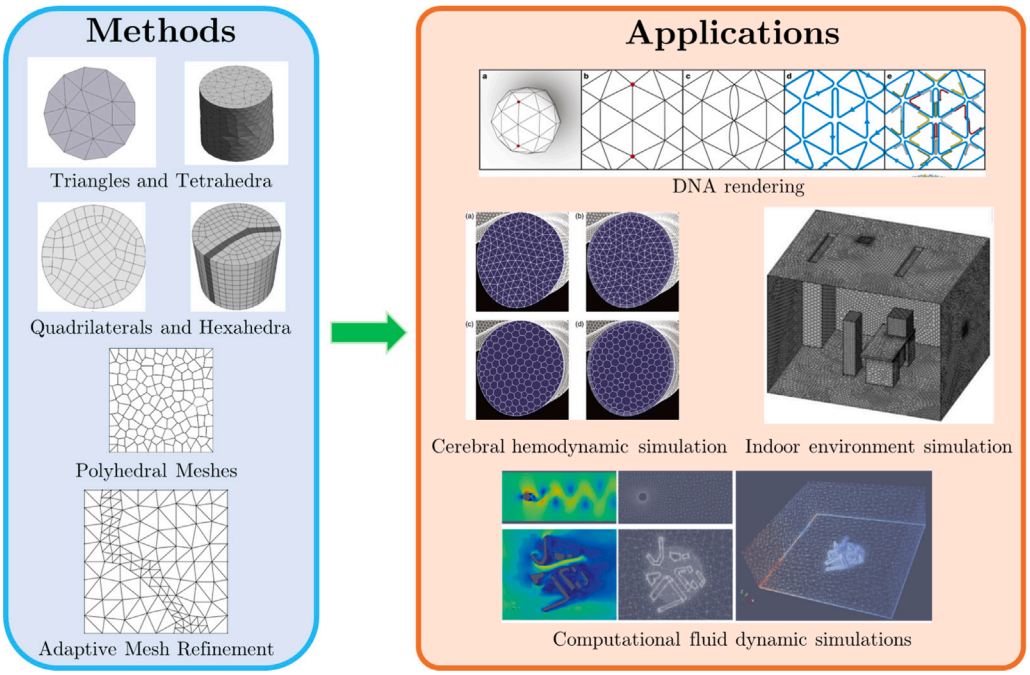


Fig. 2. Methods and applications of unstructured meshes.

Developed from SVD techniques, POD aims to identify an optimal least squares subspace for approximating data sets. In this method, we assumed that any variable $\mathbf{X} \in \mathbb{R}^{N_s \times N_h}$ could be expressed as:

$$\mathbf{X} = \sum_{i=1}^{N_h} \alpha_i \phi_i, \tag{1}$$

where N_s denotes the total number of snapshots and N_h denotes the general number of nodes on meshes. α_i denotes the ith POD coefficient and ϕ_i denotes the ith POD basis function.

The POD basis functions are obtained through SVD applied on discredited numerical solutions $\mathbf{X}$, which could be shown as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^1 \\ \mathbf{x}^2 \\ \dots \\ \mathbf{x}^{N_s} \end{bmatrix}. \tag{2}$$

The SVD computed on $\mathbf{X}$ forms:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{3}$$

where orthogonal matrix $\mathbf{U} \in \mathbb{R}^{N_s \times N_s}$, $\mathbf{V}^* \in \mathbb{R}^{N_h \times N_h}$ contains the left and right singular vectors respectively, matrix $\mathbf{\Sigma} \in \mathbb{R}^{N_s \times N_h}$ denotes the singular values of $\mathbf{X}$ and these values are listed by their magnitude. In the matrix $\mathbf{U}$, the columns represent the POD basis functions or modes, with the singular values indicating the importance of each basis function. The square of these singular values signifies the energy captured by each basis function. Following this truncation, an approximation $\tilde{\mathbf{X}}$ (using the first $\tilde{n}$ POD basis) of the variables $\mathbf{X}$ could be calculated by truncated matrices and expressed as a linear combination of the retained POD basis functions:

$$\tilde{\mathbf{X}} = \mathbf{X} \cdot \tilde{\mathbf{V}} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^* \cdot \tilde{\mathbf{V}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}} = \sum_{j=1}^{\tilde{n}} \alpha_j \phi_j, \tag{4}$$

where truncated matrices $\tilde{\mathbf{U}} \in \mathbb{R}^{N_s \times \tilde{n}}$, $\tilde{\mathbf{V}}^* \in \mathbb{R}^{\tilde{n} \times N_h}$ and $\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{\tilde{n} \times \tilde{n}}$ are used to create a reduced-order dataset $\tilde{\mathbf{X}}$ that maintains the most significant patterns in the data $\mathbf{X}$. This process is illustrated on flow past a cylinder case simulations in Fig. 3.

Moreover, Dynamic mode decomposition (DMD) is also developed for ROMs based on the SVD technique [72,73]. It is defined to identify low-order dynamics, providing insights into the system evolution over time [74]. SVD in DMD serves as a dimensionality reduction and feature extraction tool to identify the dominant spatial and temporal patterns in complex systems [75,76]. Furthermore, DMD was developed to connect to the underlying nonlinear dynamics through Koopman operator theory [77] by Mezić [78,79], and Rowley [80]. In a general DMD algorithm, we assume that data are generated by linear dynamics:

$$\mathbf{x}^{t+1} = \mathbf{A}\mathbf{x}^t, \tag{5}$$

where an operator $\mathbf{A}$ is assumed to exist and approximate the dynamics. The DMD modes and eigenvalues are intended to approximate the eigenvectors and eigenvalues of $\mathbf{A}$. For DMD on the selected variable, the discredited numerical solution is split into two matrices:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}^1 & \mathbf{x}^2 & \dots & \mathbf{x}^{N_s-1} \\ | & | & & | \end{bmatrix}, \ \bar{\mathbf{X}} = \begin{bmatrix} | & | & & | \\ \mathbf{x}^2 & \mathbf{x}^3 & \dots & \mathbf{x}^{N_s} \\ | & | & & | \end{bmatrix}, \tag{6}$$

as a set of pairs $\left\{\left(\mathbf{X}^k, \bar{\mathbf{X}}^k\right)\right\}_{k=1}^{N_s-1}$. The SVD of $\mathbf{X} \in \mathbb{R}^{N_h \times (N_s-1)}$ is computed by Eq. (3) as:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{7}$$

where $\mathbf{U} \in \mathbb{R}^{N_h \times N_h}$, $\mathbf{V} \in \mathbb{R}^{(N_s-1) \times (N_s-1)}$, $\mathbf{\Sigma} \in \mathbb{R}^{N_h \times (N_s-1)}$. A low-order approximation of $\mathbf{X}$ can be made by retaining only the first $\tilde{n}$ singular values:

$$\mathbf{X} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*, \tag{8}$$

where $\tilde{\mathbf{U}} \in \mathbb{R}^{N_h \times \tilde{n}}$, $\tilde{\mathbf{V}}^* \in \mathbb{R}^{\tilde{n} \times (N_s-1)}$, $\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{\tilde{n} \times \tilde{n}}$. Then the approximation of the dynamic matrix $\mathbf{A}$ in the low-dimensional space, $\tilde{\mathbf{A}} \in \mathbb{R}^{\tilde{n} \times \tilde{n}}$, is defined as:

$$\tilde{\mathbf{A}} = \tilde{\mathbf{U}}^* \bar{\mathbf{X}} \tilde{\mathbf{V}} \tilde{\mathbf{\Sigma}}^{-1} \tag{9}$$

The eigenvalues Λ and eigenvectors $\boldsymbol{\omega}$ of $\tilde{A}$ are computed by:

$$\tilde{\mathbf{A}}\boldsymbol{\omega} = \boldsymbol{\omega}\Lambda, \tag{10}$$

where Λ is a diagonal matrix with DMD eigenvalues $\lambda_{\tilde{n}}$ then the corresponding DMD mode is given with $\tilde{\mathbf{U}}$:

$$\boldsymbol{\Phi} = \tilde{\mathbf{U}}\boldsymbol{\omega}, \tag{11}$$

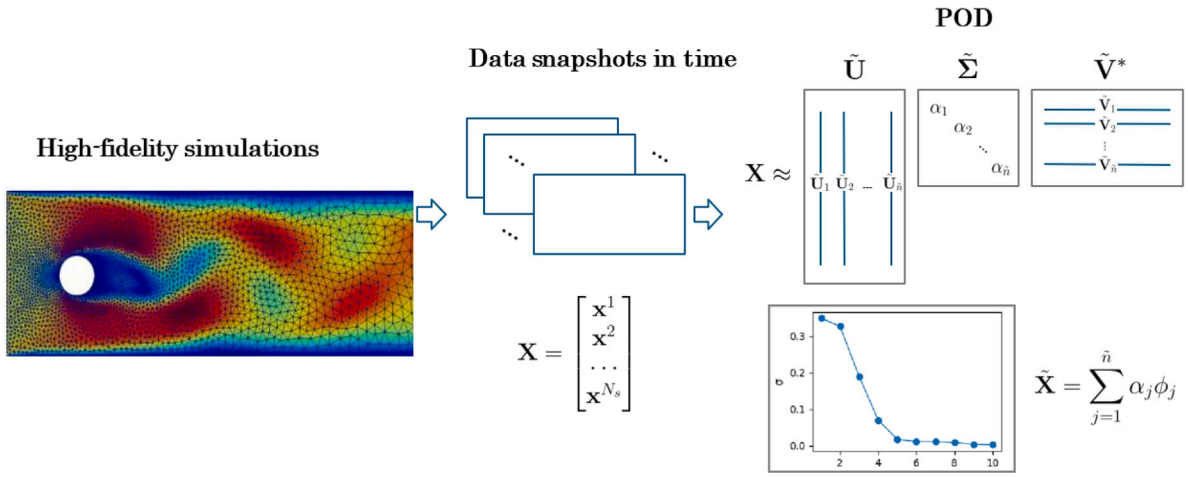


Fig. 3. POD of high-fidelity simulation data over time.

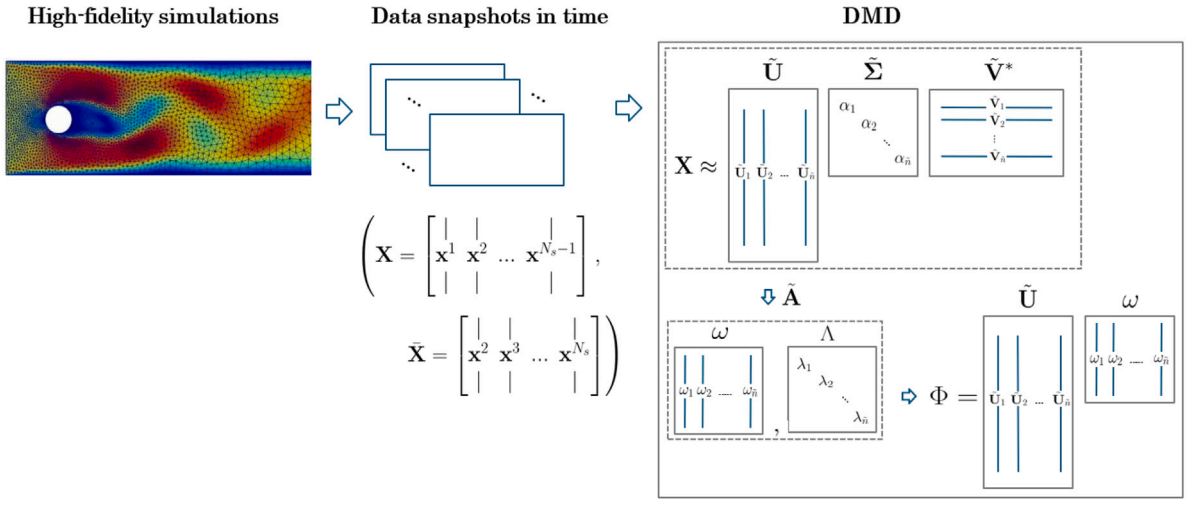


Fig. 4. DMD of dynamic modes in high-fidelity simulation data.

where $\Phi \in \mathbb{R}^{N_h \times \tilde{n}}$ contains the eigenvectors or dynamic modes corresponding to the DMD eigenvalue Λ in the original high-dimensional space. The whole DMD process on flow past a cylinder case is shown in Fig. 4.

In addition, ML tools have been developed and shown great potential in representing complex nonlinear mapping. With strong nonlinear fitting ability, ML models are implemented in ROMs' modal decomposition and evolution regression parts [81,82]. For the establishment of the appropriate coordinate system in latent space, the deep learning-based AE network has been widely used as a nonlinear model reduction alternative to linear methods, like POD, capturing features more efficiently [83]. A typical autoencoder is a self-supervised neural network with identical inputs and outputs. It includes an encoder $\mathcal{E}$ to map the inputs x into the reduced latent space and a decoder D to reconstruct the high-fidelity simulations from the latent representations $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}} = \mathcal{E}(\mathbf{x}) \quad \text{and} \quad \mathbf{x}^{\mathrm{AE}} = D(\tilde{\mathbf{x}}). \tag{12}$$

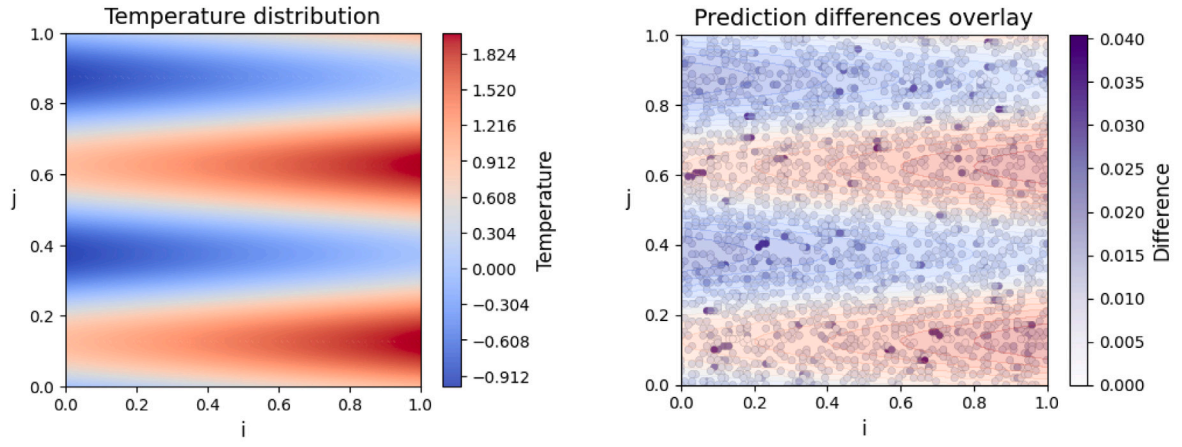
The encoder $\mathcal{E}$ and decoder D are optimised simultaneously for minimising the reconstruction loss function:

$$L_{\mathcal{E},D}(\mathbf{x}) = \|\mathbf{x} - D(\mathcal{E}(\mathbf{x}))\| = \left\|\mathbf{x} - \mathbf{x}^{\mathrm{AE}}\right\|, \tag{13}$$

where $\|.\|$ is the Euclidean norm. A series of algorithms based on AE have been developed to construct ROMs for latent-space learning [84–86].

However, a limitation of these ROMs is their lack of awareness of points position. This means that while they efficiently reduce the dimensionality of data by capturing dominant features, they might overlook spatial dependencies and interactions critical in the mesh structure [54,87]. To address this limitation, additional strategies could be integrated into ROMs, such as spatial tagging of modes or coupling with spatially-aware algorithms, to enhance the position sensitivity of these techniques. Meanwhile, these reduced-order methods inherently require a fixed mesh configuration. They operate under the assumption that the number of mesh points remains constant and that these points are consistently positioned throughout the dataset [88]. This limitation can be particularly restrictive in fields like fluid dynamics or structural mechanics, where adaptive unstructured meshes are often essential to capture complex behaviours efficiently. The integration of ROMs with adaptive mesh strategies is being explored to overcome these challenges, such as Adaptive Mesh Refinement [89,90] and re-meshing techniques [91].

Once an appropriate coordinate system is established, various ML algorithms are utilised to model the dynamics evolution for forecasting, especially in Non-Intrusive Reduced Order Modellings [62]. RNNs (see Section 2.4) have improved the modelling of temporal dependencies in ROMs [92,93], such as Long short-term memory (LSTM) networks excel in capturing long-range time dependencies, addressing the vanishing gradient issue [70,94]. Self-attention algorithm-based methods like Transformers and its variants have been explored in ROM, facing challenges in computational efficiency [95,96]. Approaches like



(a) Simulated temperature: $x_{i,j} = \mathrm{i} + \sin(4\pi * \mathrm{j})$ (b) Prediction results: Difference $= |\text{Prediction} - x_{i,j}|$

Fig. 5. Illustration of a regression problem with irregular data points using decision tree.

Sparse identification of nonlinear dynamics (SINDy) efficiently derive dynamic system models from data [97–100]. Enhanced accuracy in ROM predictions is achievable through specialised physical knowledge integration by PINN as physical-informed ROMs [101,102]. While ML models combined with ROM reducing computational loads in system modelling, the prediction process incurs deviation from the physical model due to error accumulation over successive predictions.

2.3. Shallow machine learning for dynamical systems

Conventional ML applications are increasingly pivotal in the analysis and understanding of dynamic systems and unstructured mesh environments. These systems, characterised by their complexity and ever-changing nature, present unique challenges for computational modelling and simulations. Several well-established ML algorithms discussed herein have been effectively applied to address issues pertaining to dynamical systems.

The K-Nearest neighbours (KNN) algorithm [103,104] is a simple yet effective method for classification and regression. It operates on the principle of selecting a predetermined number of neighbours (k) to determine the categorisation or value of new data points, measured using metrics such as Euclidean, Manhattan and Minkowski [105]. KNN finds considerable application in modelling the dynamics of physical systems, where it can analyse spatial or temporal relationships among data points to simulate complex behaviours [106]. A notable implementation is the KNN-DMD [107], which utilises KNN to select and average the closest k DMD solutions (introduced in Section 2.2). This is based on the distances between the parameters of interest and other parameters, thereby adapting DMD to parametrised problems effectively.

KNN's main challenges are high computational demands with large datasets [107] and difficulties handling chaotic time-series data [108,109]. Research continues to improve its efficiency and adaptability [107,109].

A decision tree [110] is a binary tree model used to handle unstructured data [111] and dynamic systems [112], making it ideal for classification (categorising phenomena) and regression (predicting values, as shown in Fig. 5). The hierarchical nature of this model with its nodes and directed edges, offers clear interpretability [113–115], which is a critical feature in physics that allows for the detailed tracing of the decision-making process [116].

Further advancing on the decision tree's capabilities, the RF [117] model emerges as a robust ensemble technique designed to enhance predictive accuracy and prevent overfitting. This ensemble method proves particularly potent in dynamic systems, offering enhanced performance and adaptability [106,118,119]. eXtreme gradient boosting (XGBoost) [120], on the other hand, marks a significant evolution in the realm of ensemble learning, specifically advancing the use of boosting techniques. Unlike RF approach, which constructs an ensemble of decision trees in parallel, XGBoost builds its tree models sequentially. Each tree in the sequence is designed to address and correct the errors of its predecessors, leading to progressively improved model accuracy. XGBoost has demonstrated exceptional effectiveness in capturing complex relationships within datasets [121,122] and performing reliably across various interpolation scenarios [123].

A Gaussian Process (GP) [124] is a probabilistic model for continuous domains that represents any set of points as a multivariate Gaussian distribution (Fig. 6). Defined by a mean function μ and a covariance function σ^2, GPs model distributions over functions, enabling flexibility in regression and classification for dynamical systems [125]. Their ability to handle nonlinear relationships stems from the kernel, which encodes assumptions like smoothness or periodicity. By optimising kernel hyperparameters and employing non-stationary or autoregressive formulations [126,127], GPs adapt to complex, high-dimensional dependencies, such as time-varying noise or evolving dynamics, without rigid parametric constraints.

Like KNN, RF, XGBoost, and Gaussian Process models are widely used with reduced-order models to capture nonlinear dynamics in reduced spaces [129–133]. While effective for dynamical systems on unstructured grids, tree models are prone to overfitting in high dimensions and are sensitive to noise, while GP models face limitations like the assumption of stationarity and challenges with high-dimensional systems, complicating model construction and inference [134,135].

2.4. Convolutional and recurrent neural networks

Deep neural networks are machine-learning techniques that are capable of handling high-dimensional, nonlinear dynamical systems, and automatically learn complicated patterns from data without extensive manual feature engineering, which are complex to conventional methods as described in Section 2.3, especially for large data inputs. There are various types of neural networks each designed for a specific task; however, CNN [136,137] and RNN [138] will be briefly discussed in this section regarding their capability of handling spatial–temporal systems.

CNNs are widely used in analysing high-dimensional spatial data such as images [136,137,139] and dynamical systems like identifying patterns or features in spatially distributed data [140–142]. In dynamical systems with sensor data inputs, CNNs can be leveraged for feature extraction and pattern recognition. For instance, in environmental monitoring systems, CNNs can help in identifying anomalies or predicting

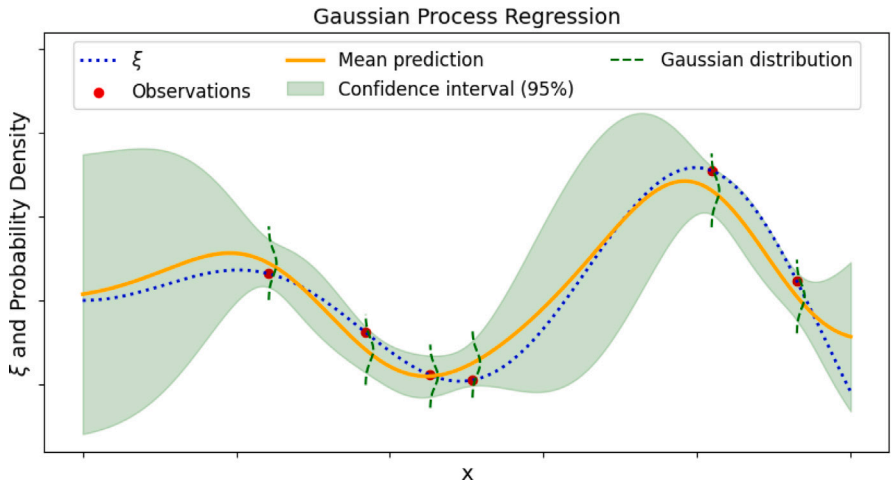


Fig. 6. Gaussian process regression (as known as Kriging [128]) to predict complex, nonlinear relationships.

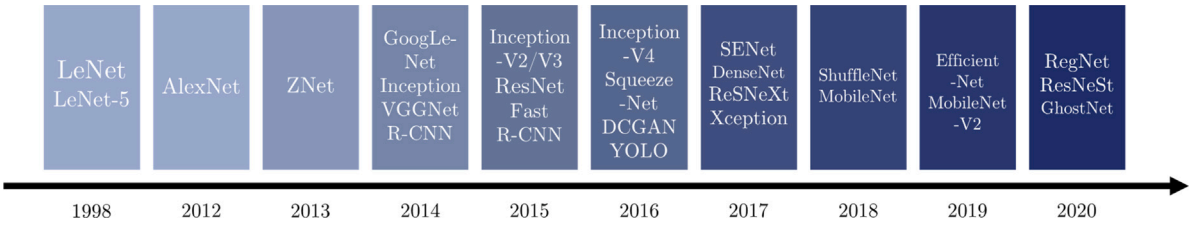


Fig. 7. Evolutionary trend of classic CNN models: Visual Geometry Group Network, Regional-Based Convolutional Neural Network, ResNet, Deep Convolutional Generative Adversarial Network, Squeeze-And-Excitation Network.

future states based on sensor readings [140]. The architecture of CNN has undergone a significant transformation, as illustrated in 7, with 2017 marking a notable surge in the development of CNN models.

CNNs were designed originally to process structured data, particularly images [139] due to their ability to capture hierarchies through convolutional layers. However, through advancements in research and engineering, CNNs have been extended to handle unstructured data [14,143], like text, audio, and time-series data. Residual Network (ResNet) [144] is a specialised CNN that employs skip connections to learn residual functions, addressing the vanishing gradient problem and enabling the training of deeper networks. Improved Residual Networks (iResNet) [145] further enhance information flow, optimise shortcuts, and improve spatial feature learning, allowing for extremely deep networks without added complexity [146]. Skip connections are also an essential component of transformer type neural networks which could effectively handle irregular data as detailed in Section 3.3.

The significance of sequential data processing and time series forecasting has grown substantially across various domains, encompassing finance, economics, weather prediction, and natural language processing. Within this landscape, RNNs [138,147,148], notably LSTM networks, have risen to prominence as a favourite approach for managing sequential data and predicting time series. The fundamentals of RNN are presented in [138] utilising differential equations encountered in many branches of science and engineering. The canonical formulation of RNN by sampling delayed differential equations can be applied in modelling of complex processes in physics. Chen et al. [149] proposes pyramid convolutional RNN for MRI image reconstruction based on low, middle, and high-frequency information in a sequential pyramid order with better recovery of fine details different from CNN which learns the three frequency categories individually. The more robust LSTM, a highly used type of RNN overcomes the challenges of the standard RNN, and is capable of representing high-dimensional and complex systems, which are typically difficult to model using conventional machine learning techniques [138,150]. For instance, RNN and LSTM are used in capturing dynamical systems with multivariate parameters optimisation to minimise errors [151]. Blandin et al. [152] use a multi-variate LSTM to predict rapidly changing geomagnetic conditions electric fields form within the Earth's surface and induce currents. A Convolutional LSTM network provides a fast and affordable prediction of spatio-temporal systems in expensive computation fluid solvers [153].

The main models for handling sequential data use advanced recurrent or convolutional neural network that incorporate an encoder and a decoder [21]. Complex dynamic systems with multiple time lags are regarded as high-dimensional dynamical systems with time lags and the temporal dependence plays a key role in modelling them. However, in classical CNN models, the number of operations needed to relate signals of two arbitrary input or output positions grows in the distance between positions, creating complexities in learning dependencies between distant positions. RNNs appear as ideal candidates to model, analyse, and predict complex and dynamical systems due to their temporal occurrence. However, classical RNNs do not carry out sequential reasoning, a process based on attention [21,154]. Position embedding technique in attention mechanism incorporates information about position of tokens within a sequence that conventional RNN and CNN are unable to capture. It has the capabilities of handling sequences of variable length without tempering performance and hence is able to handle sparse, unstructured, and missing data. More details about the attention mechanism and the transformer type neural networks are given in Section 3.3 of this review paper.

2.5. Gridding irregular data via interpolation

Discretisation [44,155] constitutes an indispensable step for numerically resolving Partial differential equations (PDEs) that govern the evolution of dynamical systems. In this process, continuous physical fields in the space–time domain are transformed into discrete representations by meshing the domain into thousands or even millions of small

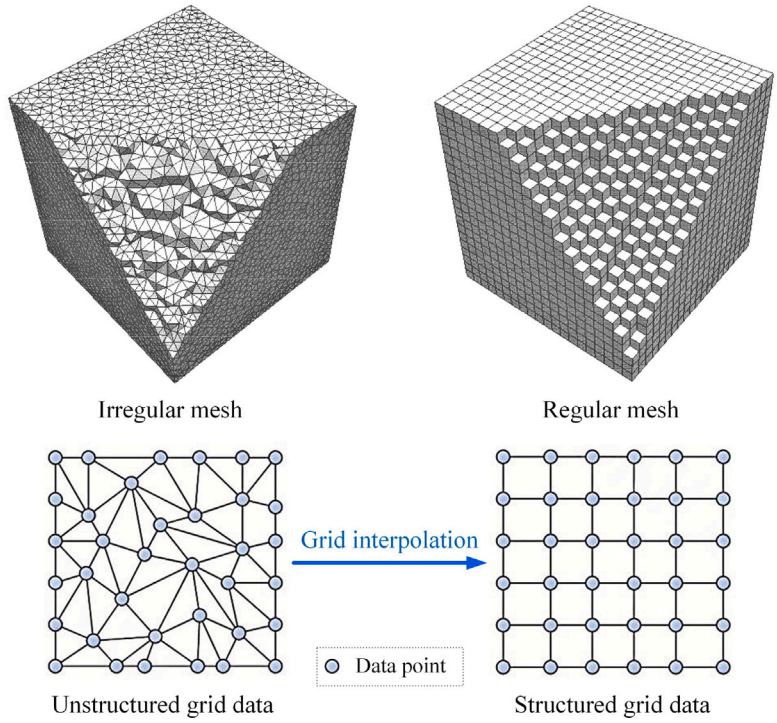


Fig. 8. Illustration of converting unstructured grid data to structured grid data via interpolation.

elements, making the problem computationally solvable. Generally, meshing can be categorised into regular and irregular types, as depicted in Fig. 8. While regular meshes facilitate computer programming, irregular meshes are far more prevalent for approximating complex, arbitrary geometrical shapes in practical applications.

With the increasing use of computational intelligence across various fields, there is a growing demand to convert large volumes of unstructured data into structured formats that can be processed by machine and deep learning models. This is particularly important because many models, including perceptrons and convolutional neural networks, are usually optimised to work with data arranged in a regular Cartesian grid as mentioned in Section 2.4. To meet this need, interpolation methods are commonly employed to convert irregularly gridded data into a raster dataset composed of regularly spaced square pixels, enabling further analysis and model training in machine learning applications. Commonly used interpolation methods include nearest neighbour interpolation, linear or barycentric interpolation, radial basis function interpolation, and Kriging interpolation.

Nearest neighbour interpolation [104,156] estimates the value of a variable at any given location by using the value of the nearest known data point. The concept of Voronoi tessellation [157] can be employed to measure the distance between points and determine proximity, as illustrated in Fig. 9a. In this method, the entire domain is subdivided into cells based on the locations of the known data points, with each cell assigned the value of its corresponding data point. This means that the value of each known data point is assigned to all unknown data points within its associated cell. While nearest neighbour interpolation is computationally efficient due to its simplicity, it typically results in lower interpolation accuracy, especially when the data points are sparsely distributed or exhibit significant spatial variation.

Linear interpolation [158,159] creates new data points within the range of known data points by fitting a linear polynomial. For the 1D case, given a field vector $\mathbf{x}$ and the values of two know data points x_{i_0} and x_{i_1}, the linear interpolation polynomial x_i within the range of $i \in [i_0, i_1]$ is mathematically expressed as:

$$x_i = \frac{i_1 - i}{i_1 - i_0} x_{i_0} + \frac{i - i_0}{i_1 - i_0} x_{i_1} . \tag{14}$$

This univariate interpolation can also be straightforwardly extended to multivariate interpolation on 2D and 3D regular grids, called bilinear and trilinear interpolation, respectively. Barycentric interpolation [160,161] generalises linear interpolation for arbitrary (non-regular) grids, allowing transformation of unstructured grid data to structured grid data. For the 2D case, barycentric interpolation creates new data points from three near-neighbours that form a triangle, as depicted in Fig. 9b. Given the values of three vertices of the triangle $x_{(i_0,j_0)}$, $x_{(i_1,j_1)}$ and $x_{(i_2,j_2)}$, the value of an unknown data point (i, j) within this triangle can be estimated via weighted average:

$$x_{(i,j)} = w_0\ x_{(i_0,j_0)} + w_1\ x_{(i_1,j_1)} + w_2\ x_{(i_2,j_2)} , \tag{15}$$

where w_0, w_1 and w_2 are the barycentric weights, which can be calculated from the coordinates of the three data points, given by:

$$\begin{aligned} w_0 &= \frac{(j_1 - j_2)(i - i_2) + (i_2 - i_1)(j - j_2)}{(j_1 - j_2)(i_0 - i_2) + (i_2 - i_1)(j_0 - j_2)} , \\ w_1 &= \frac{(j_2 - j_0)(i - i_2) + (i_0 - i_2)(j - j_2)}{(j_1 - j_2)(i_0 - i_2) + (i_2 - i_1)(j_0 - j_2)} , \\ w_2 &= 1 - w_0 - w_1 . \end{aligned} \tag{16}$$

These weights, often called barycentric coordinates, are crucial for accurately estimating values at unknown locations based on the positions and values of the known data points.

Radial basis function interpolation [162,163] represents a sophisticated technique for constructing high-order accurate interpolants from unstructured data. In this method, the interpolant is expressed as a weighted sum of radial basis functions. Given a set of data points consisting of $\left(\mathbf{i}_k, x_{\mathbf{i}_k}\right)$ for $k = 1, 2, \ldots, n$, where $\mathbf{i}_k$ denotes the vector of

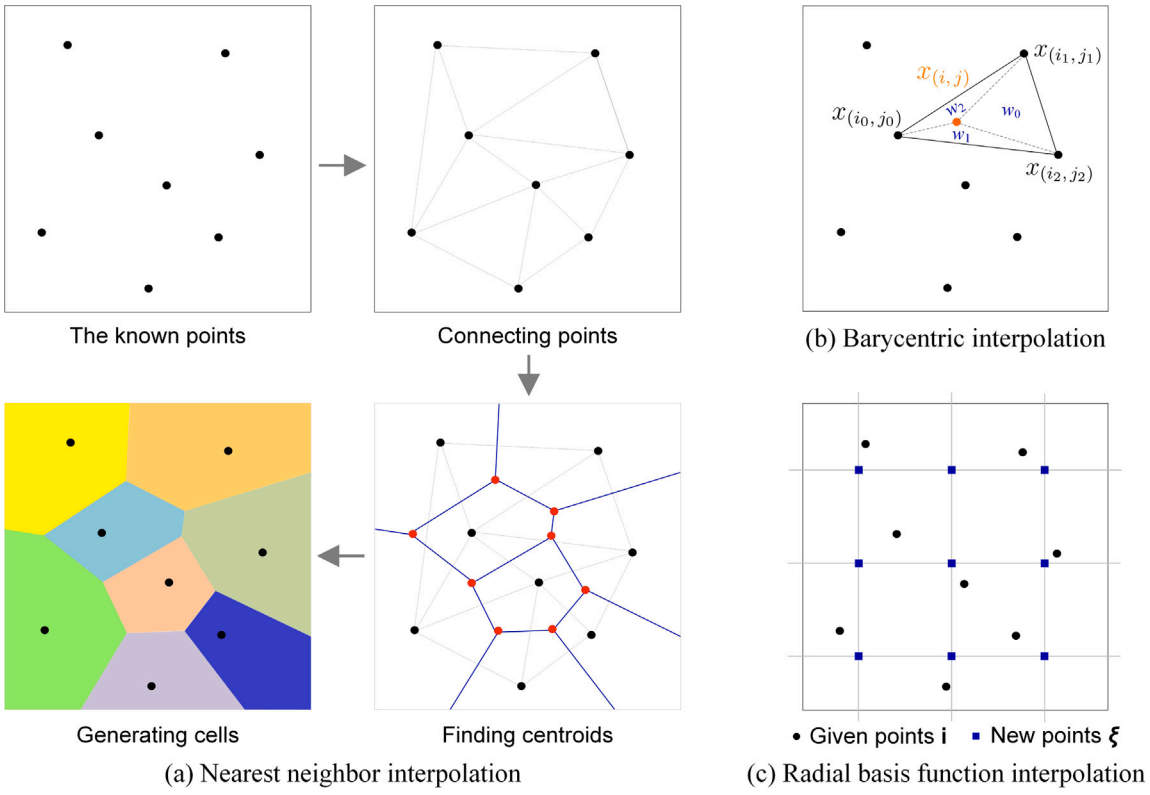


Fig. 9. Graphical illustration of unstructured grid interpolation.

coordinates, the goal of radial basis function interpolation is to find an interpolant $s(\mathbf{i}_k)$, satisfying

$$s(\mathbf{i}_k) = x_{\mathbf{i}_k}, \quad k = 1, 2, \dots, n. \tag{17}$$

For any point $\mathbf{i} \notin \{\mathbf{i}_k\}_{k=1,2,\dots,n}$, the interpolant $s(\mathbf{i})$ is expressed as a linear combination of radial basis functions $\phi(\mathbf{i})$:

$$s(\mathbf{i}) = \sum_{k=1}^{n} w_k \phi\big(\|\mathbf{i} - \mathbf{i}_k\|\big), \quad \mathbf{i} \in \mathbb{R}^d. \tag{18}$$

From the condition in Eq. (17), the following system of equations for the weights w_k:

$$\sum_{k=1}^{n} w_k \phi\big(\|\mathbf{i}_q - \mathbf{i}_k\|\big) = x_{\mathbf{i}_k}, \quad q \neq k, \tag{19}$$

where $\mathbf{i}_q$ and $\mathbf{i}_k$ are the vectors of coordinates of different known data points.

By solving the above system of linear equations, the unique solution for the weight vector $\boldsymbol{w} = [w_1, w_2 ..., w_n]$ can be obtained. The unknown value x_ξ at a point $\boldsymbol{\xi}$ on the structured grid can then be estimated as

$$x_\xi = s(\xi) = \sum_{k=1}^{n} w_k \phi\big(\|\boldsymbol{\xi} - \mathbf{i}_k\|\big), \quad \boldsymbol{\xi} \in \mathbb{R}^d, \tag{20}$$

as illustrated in Fig. 9c.

Kriging [164,165] is a spatial interpolation technique employed to derive predictions at unsampled locations based on observed geostatistical data. Given a set of observation points represented as $(\mathbf{i}_k, x_{\mathbf{i}_k})$ for $k = 1, 2, \dots, n$, Kriging interpolation estimates the value at an unobserved location $\boldsymbol{\xi}$ through a weighted average:

$$\hat{x}_\xi = \sum_{k=1}^{n} w_k x_{\mathbf{i}_k}, \quad k = 1, 2, \dots, n. \tag{21}$$

This estimation minimises the mean squared prediction error over the entire state space, defined as:

$$E\big[(\hat{x}_\xi - x_\xi)^2\big]. \tag{22}$$

The unknown Kriging weights $\boldsymbol{w} = [w_1, w_2 ..., w_n]$ in Eq. (21) can be derived from the estimated spatial structure of the known dataset. Specifically, these weights are obtained by fitting a variogram model to the observed data, which elucidates how the correlation between observation values varies with distance between locations.

Once the Kriging weights are obtained, they are applied to the known data values at observed locations to compute predicted values at unobserved locations, as illustrated in Fig. 10. These weights reflect the spatial correlation inherent in the data, accounting for both geographical proximity and similarity among data points. Consequently, observed locations that exhibit stronger correlation and are closer to prediction sites receive greater weight compared to those that are uncorrelated and/or more distant. Additionally, the weighting scheme considers the spatial arrangement of all observations; thus, clusters of observations in oversampled areas carry less weight due to their lower information content relative to single locations. Under certain assumptions, Kriging predictions serve as best linear unbiased estimators. There exist various types of Kriging methods differentiated by their underlying assumptions and analytical objectives [164,165]. For instance, Simple Kriging [166] presumes that the mean $\mu(\mathbf{i})$ of the random field is known; Ordinary Kriging [167] assumes an unknown but constant mean $\mu(\mathbf{i}) = \mu_0$; while Universal Kriging [168] is applicable for datasets characterised by an unknown non-stationary mean structure.

Other interpolation methods, such as those based on polynomial functions, finite element basis functions [169], or splines [170], have also been widely adopted for unstructured data. Their combinations with ML will be further discussed in Section 3.1.

3. Machine learning models designed for unstructured grid data

In this section, we review machine learning models, focusing on neural network structures and specific preprocessing methods that can be applied to dynamical systems with unstructured grid data. These include architectures such as specific CNNs, GNNs, and transformers.

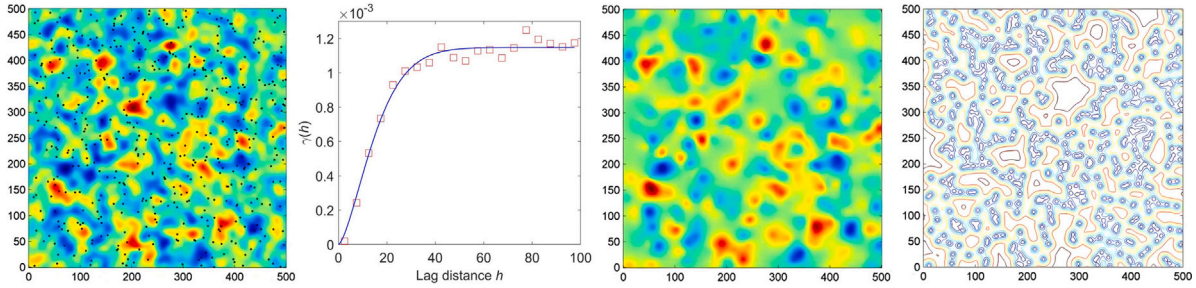


Fig. 10. Graphical illustration of Kriging interpolation.

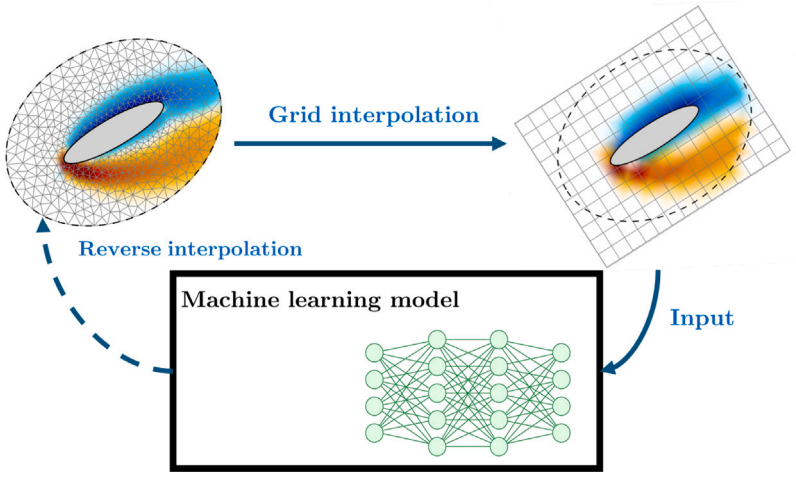


Fig. 11. Illustration of ML methods with interpolated inputs for unstructured data. The figure is based on elements in Figure 9 of [180].

3.1. Machine learning with preprocessing

3.1.1. Convolutional neural network with interpolation methods

Traditional ML approaches, in particular, CNN-based methods are limited to structured grid data and can only process fixed-size input data when an MLP layer is included in the neural network architecture. These limitations restrict their applicability to real-world problems, where incomplete or sparse observations and time-varying sensors are common [171,172]. Significant effort has been dedicated to integrating machine learning with interpolation methods to address the challenges of data sparsity and irregularity (see review papers [173,174]).

Sparse and continuous CNNs have been developed to handle sparse and irregular data points in convolutional operations [175–177]. Some pioneering works have applied these advanced CNN approaches in CFD with unstructured grid data. For instance, Wu et al. [178] use a sparse CNN model for end-to-end prediction of supersonic compressible flow fields around airfoils from spatially sparse geometries, while Wen et al. [179] adapt continuous CNNs for irregularly placed multiphase flow particles. However, sparse CNNs involve specialised algorithms for handling sparse data, leading to increased computational overheads and optimisation difficulties. Continuous CNNs, designed for continuous input spaces, can encounter discretisation errors and require more computational resources for precise computations [176]. Consequently, their application in high-dimensional dynamical systems, such as numerical weather prediction or fine-resolution CFD, remains limited.

Another common strategy to address the challenge of unstructured data involves converting unstructured grid data into structured grid data, often resizing it to a fixed dimension before feeding it into a machine learning model, as shown in Fig. 11. Typically, Hou et al. [181] and Cao et al. [182] combined Kriging interpolation with CNN and RNN for spatially irregular input data. Following a similar idea, the recent work by Fukami et al. [15] leverages Voronoi tessellation, as introduced by [183], to overcome the challenges posed by sparse observations and the varying number of sensors in CNN-based field reconstruction. Their method considers a set of observable points (sensors) at a given time, located at $\{(i_k, j_k)\}_{k\in\{1,\dots,k*\}}$ where

$$\{i_k, j_k\} \in [1, \dots, N_x] \times [1, \dots, M_x] \quad \text{and} \quad k* \text{ is the number of sensors.} \tag{23}$$

Suppose x_k is the observed value at $\{i_k, j_k\}$. A Voronoi cell R_k associated to the observation $\{x_k, i_{t,k}, j_{t,k}\}$ can be defined as

$$R_k = \Big\{\{i_r, j_r\} \;\Big|\; d((i_r, j_r), (i_k, j_k)) \le d((i_r, j_r), (i_q, j_q)), \quad \forall \;\; 1 \le q \le k^* \quad \text{and} \quad q \ne k\Big\}. \tag{24}$$

Here, $d(\cdot)$ is the Euclidean distance. Therefore, the state space can be partitioned into several Voronoi cells regardless of the number of sensors, that is,

$$\mathcal{U}_{[1,\dots,N_x]\times[1,\dots,M_x]} = \bigcup_{k=1}^{k*} R_k \quad \text{and} \quad R_k \cap R_q = \emptyset \quad (\forall \quad k \ne q), \tag{25}$$

where $\mathcal{U}_{[1,\dots,N_x]\times[1,\dots,M_x]}$ designates the full discretised space. A tessellated observation $\hat{\mathbf{x}} = \{\hat{x}_{i_x,j_x}\} \in \mathbb{R}^{N_x \times M_x}$ in the full state space can be obtained by

$$\hat{x}_{i_x,j_x} = x_k \quad \text{if} \quad (i_x, j_x) \in R_k. \tag{26}$$

Once the tessellated observation is obtained, regression ML models could be implemented to perform field reconstruction, prediction of future time steps or parameter calibration.

The Voronoi tessellation-assisted CNN has demonstrated superior accuracy and efficiency in CFD and geoscience datasets compared to

traditional interpolation techniques like Kriging [100,184]. The recent work by [16] integrates Voronoi tessellation-assisted CNN into a variational data assimilation framework, enhancing prediction accuracy using the same unstructured and sparse observation data.

On the other hand, the work of [185] incorporates Kriging interpolation into the loss function calculation for unstructured data. This approach differs from previously mentioned methods by explicitly accounting for interpolation errors during the training phase. A differentiable nearest neighbour algorithm has also been proposed by [186]. Although it has primarily been tested on image classification and restoration tasks, this method shows promise for enhancing interpolation in unstructured grid systems.

ML techniques have also been developed to perform interpolation and super-resolution directly on sparse or unstructured grid data [3]. Typical examples in computational physics include airfoil wake and porous flow [187,188]. Furthermore, Kashefi et al. [189] leveraged PointNet architecture [190] to predict flow fields in irregular domains by treating CFD grid vertices as point clouds, preserving the accuracy of unstructured meshes without data interpolation. The approach accurately represents object geometry, maintains boundary smoothness, and predicts flow fields much faster than conventional CFD solvers, while generalising well to unseen geometries. Compared to conventional interpolation methods such as Kriging or nearest neighbour, which require strong mathematical assumptions, ML-based approaches offer more flexible and non-parametric interpolations for irregular data [191].

3.1.2. Machine learning with mesh reordering and transformation

As mentioned in Section 2.2, conventional linear projection methods for ROM, such as POD and DMD, can be seamlessly applied to unstructured data by flattening the entire physics field into a single column vector. This approach simplifies the integration of machine learning models for tasks like field prediction or parameter identification. For instance, Xiao et al. [62] and Casenave et al. [125] apply POD to unstructured grid data of dynamical systems, followed by a Gaussian process to model the dynamics of air pollution within a reduced-order space. Similar approaches have also been applied to nuclear reactor physics [192] and unsteady flow simulations [193].

Although POD-type methods can be directly applied to unstructured grid data, they are less efficient than deep learning techniques, particularly autoencoders, for handling significantly nonlinear dynamical systems. Building on the concepts of POD and DMD, researchers have explored the use of autoencoders with one-dimensional CNNs by first flattening the physical field into a single column vector. Since the one-dimensional CNNs uses a fixed-size filter to capture the pattern and correlation in input vectors, the ordering of unstructured grids plays a key role in the performance of such approaches [14,194]. In [195], the authors utilise the Cuthill–McKee algorithm [196], which draws on concepts from graph theory, to reorder grid points in a way that minimises the bandwidth of the adjacency matrix as shown in Fig. 12(a). As a result, the reordering of grids into the same convolutional window enhances the numerical accuracy of the autoencoder [195]. In a related approach, the recent study by [14] employs space-filling curves to determine an ordering of nodes or cells that converts multi-dimensional data on unstructured meshes into a one-dimensional format, as illustrated by Fig. 12(b). This transformation allows the optimal application of 1D convolutional layers to the unstructured data. Such a reordering method with space-filling curves has also been applied to regular image data [197], where it achieves a superior performance compared to conventional CNN approaches.

In line with the principle of grid reordering, coordinate transformation can be employed to adapt standard machine learning models for non-homogeneous mesh structures. For instance, PhyGeoNet [198] utilises a coordinate transformation that maps an irregular domain onto a structured mesh space, enabling the application of convolutional operations on flow fields for fluid flow regression. Along similar lines, Chen et al. [199] select the cell lengths (both horizontal and vertical) and maximum included angle to form a three-channel quality feature, analogous to the RGB three-channel feature of images, to perform CNN for mesh quality evaluation. Another family of approaches involves transforming data into the spectral domain to address the challenges posed by complex geometries [200–202]. In these methods, the spatial coordinates of the mesh cells are typically converted into frequency coefficients within the spectral domain before being input into a machine learning model [200]. In the frequency domain, these methods apply a global convolution operation, which is a key differentiator from traditional CNNs. Instead of using local convolutions that only capture local interactions, spectral domain machine learning [203] can capture global patterns across the global input space.

There has also been an effort to merge the flexibility of linear projection techniques like POD with the ability of neural networks to manage nonlinear patterns. One common approach, known as SVD-AE or POD-AE, involves applying ML-based autoencoding to the modal coefficients derived from SVD-based methods [85,204–206]. As a result, the machine learning model can effectively manage data with complex geometries, though it requires a fixed input dimension. More precisely, both the input and output of the AE (with encoder $\mathcal{E}$ and decoder D) in SVD-AE are the compressed vectors $\tilde{\mathbf{x}}_{\text{SVD}}$ obtained through SVD, i.e.,

$$\tilde{\mathbf{x}}_{\text{SVD}} = \mathbf{U}\mathbf{x}, \quad \tilde{\mathbf{x}} = \mathcal{E}(\tilde{\mathbf{x}}_{\text{SVD}}) \quad \text{while} \quad \tilde{\mathbf{x}}^r_{\text{SVD}} = D(\tilde{\mathbf{x}}), \quad \mathbf{x}^r_{\text{SVD-AE}} = \mathbf{U}^T\, \tilde{\mathbf{x}}^r_{\text{SVD}}, \tag{27}$$

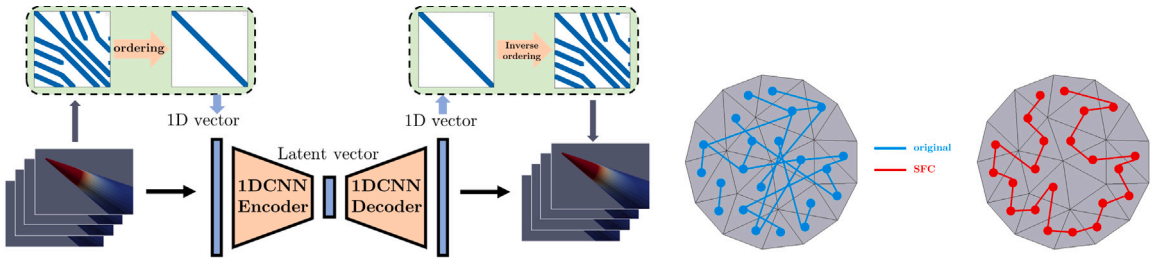
where $\tilde{\mathbf{x}}^r_{\text{SVD}}$ and $\mathbf{x}^r_{\text{SVD-AE}}$ denote the reconstruction of the POD or DMD coefficients and the reconstruction of the full physical field, respectively.

In fact, numerous studies have investigated the equivalence between POD and linear autoencoder networks [207,208], leading to the possibility of jointly training these two-stage data processing methods. The use of SVD-AE approaches extends to various applications, including nuclear reactor physics [85], multiphase flow CFD [195] and street-level air pollution estimation [204], both of which involve unstructured grid data. The latter is illustrated in Fig. 13.

3.2. Graph neural networks

GNNs are specialised in learning structured data on graphs, where nodes represent interconnected data points and edges denote the relationships between them. GNNs are particularly suitable for modelling unstructured meshes due to their inherent compatibility with such data structures. Unstructured meshes can be directly represented as graphs, with mesh vertices as nodes and connections formed by mesh edges. This allows for more efficient utilisation of training data by eliminating the need to interpolate unstructured meshes onto uniform grids, which is, as discussed in Section 3.1, a process typically required when using CNNs designed for structured grid-like datasets such as images. It should also be noted that multiple means exist for constructing a graph from an unstructured mesh. Alternatives such as defining elements or cells as nodes and creating edges based on adjacent elements or cells sharing a face, or incorporating nearest neighbour methods can be equally valid. The selection among these methods depends on the specific requirements of the modelling problem.

In the context of physical simulations, a mesh graph can be formulated as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes a set of edges defining connections between pairs of nodes whose set is $\mathcal{V}$. Node-level features can be represented by $\mathbf{V}^G \in \mathbb{R}^{|\mathcal{V}| \times N^v_f}$, where N^v_f is the number of node-level features, which typically includes spatial locations and physical (flow) field parameters measured at the corresponding points in the physical domain. The mesh graph connectivity can be described with the adjacency matrix $\mathbf{A}^G \in \mathbb{Z}^{|\mathcal{V}| \times |\mathcal{V}|}$, where an entry a_{uv} is set to one if there exists an edge from node u to node v, and zero otherwise. $\mathbf{E}^G \in \mathbb{R}^{|\mathcal{E}| \times N^e_f}$ represent edge-level features, with N^e_f indicating the number of edge-level features per edge. Moreover, it can also be beneficial to



(a) Reordering using Cuthill-McKee algorithm (b) Illustration of a space filling curve

Fig. 12. 1D CNN-based autoencoder with grid data reordering.

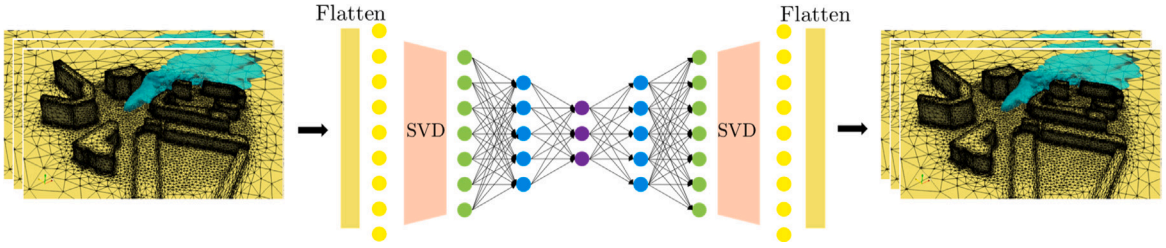


Fig. 13. SVD-AE for simulated air pollution data with unstructured meshes.

define a graph-level attribute $\mathbf{u} \in \mathbb{R}^{N_f^g}$ that contains global information pertinent to the entire graph, such as simulation parameters and inflow conditions, which are universally applicable to all nodes and edges. The final input to the GNN can be represented as the tuple $\mathbf{G} = (\mathbf{V}^G, \mathbf{E}^G, \mathbf{u}, \mathbf{A}^G)$. A node-level regression task can then be setup to use the input graph processed by layers of GNNs to predict various physical quantities at each node, such as pressure, velocity, or temperature, depending on the specific application.

While CNNs excel at capturing local patterns and maintaining translational invariance in images with convolutional filter of fixed sizes, GNNs can better adapt to the irregular connectivity structures found in unstructured meshes and can accommodate to graphs of diverse sizes and structures. GNNs function by processing and integrating information from neighbouring nodes within a graph structure. The 1-hop neighbourhood of node u can be defined as $\mathcal{N}_u = \{v|(u,v) \in \mathcal{E} \text{ or } (v,u) \in \mathcal{E}\}$, with the neighbourhood's features represented as the multiset $\mathbf{X}_{\mathcal{N}_u} = \{\{\mathbf{x}_v : v \in \mathcal{N}_u\}\}$ where $\mathbf{x}_v$ represents the feature vector of node v. GNNs aggregate features over local neighbourhoods within the graph by applying shared, permutation invariant functions $\phi(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u})$. Most graph neural networks can generally be categorised into one of the three flavours [209,210]: convolutional [211–214], attentional [215,216] and message-passing [217,218]. Each flavour determines how the features of neighbouring nodes are processed and aggregated, offering different levels of complexity for capturing the interactions across the graph. The output feature representation for a node u, denoted $\mathbf{h}_u$, varies across different GNN flavours as follows:

$$\text{Convolutional}: \mathbf{h}_u = \phi\big(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} c_{uv}\psi(\mathbf{x}_v)\big), \tag{28a}$$

$$\text{Attentional}: \mathbf{h}_u = \phi\big(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} a(\mathbf{x}_u, \mathbf{x}_v)\psi(\mathbf{x}_v)\big), \tag{28b}$$

$$\text{Message-passing}: \mathbf{h}_u = \phi\big(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\big), \tag{28c}$$

where ϕ and ψ are trainable neural networks, $\bigoplus$ is a permutation-invariant aggregation function such as mean, sum or maximum. In the convolutional flavour, the importance of a neighbouring node v on node u's feature representation is quantified by a constant c_{uv}, which is a direct function of the graph's structural connectivities. The attentional flavour, on the other hand, computes this influence through a trainable self-attention mechanism a (see Section 3.3 for details). In the message-passing flavour, ψ is a trainable function that can compute and convey arbitrary vectors or messages from node v to u. While the message-passing flavour offers more expressive and flexible modelling by computing vector-valued messages, they tend to be more memory-intensive and difficult to train. Attentional GNNs provide a more scalable middle ground by passing scalar-valued messages across edges, and convolutional GNNs are most suited for the efficient processing of homophilous graphs.

In particular, graph convolutional network (GCN), a widely used variant of GNNs, are effective at handling unstructured meshes and capturing spatial dependencies within the data. When combined with LSTM networks, which are adept at modelling temporal dependencies in sequential data, this integrated approach becomes a powerful framework for simulating dynamic fluid flow. An illustrative schematic of this method is shown in Fig. 14.

In recent years, GNNs of these different flavours have been widely applied to modelling complex computational physics problems with unstructured meshes. For instance, the work of [219] trained convolutional graph networks to accurately predict laminar flows around airfoils of different shapes. The work of [220] combined convolutional graph networks and a differentiable CFD solver that operates at a significantly coarser resolution to develop a surrogate model that can better generalise to previously unseen flow conditions. He et al. [221] developed graph neural networks that can be trained to reconstruct missing information from partial or incomplete flow fields and accurately predict fluid dynamics from sparse inputs. Li et al. [222] applied GNNs to accelerate molecular dynamics simulations, where GNNs are used to learn interactions between particles and predict the forces between atoms in molecular systems. Pichi et al. [223] introduced a graph convolutional autoencoder for ROM, which shows great physical compliance with unstructured meshes. Their recent work [224] introduced a graph feedforward network for resolution-invariant reduced-order operators to handle unstructured grid data of different resolutions, which is a key challenge in GNNs. Likewise, Liu et al. [225] developed UNet inspired multi-resolution GNNs that uses a hierarchical graph structure to capture different levels of resolution of the data in order to allow for efficient and accurate approximations of PDE solutions. Chen et al. [226] trained Convolutional Graph Networks on laminar flow around various two-dimensional shapes and the developed surrogate model showed promising results in predicting flow fields and aerodynamic properties including drag and lift. Suk et al. [227] developed

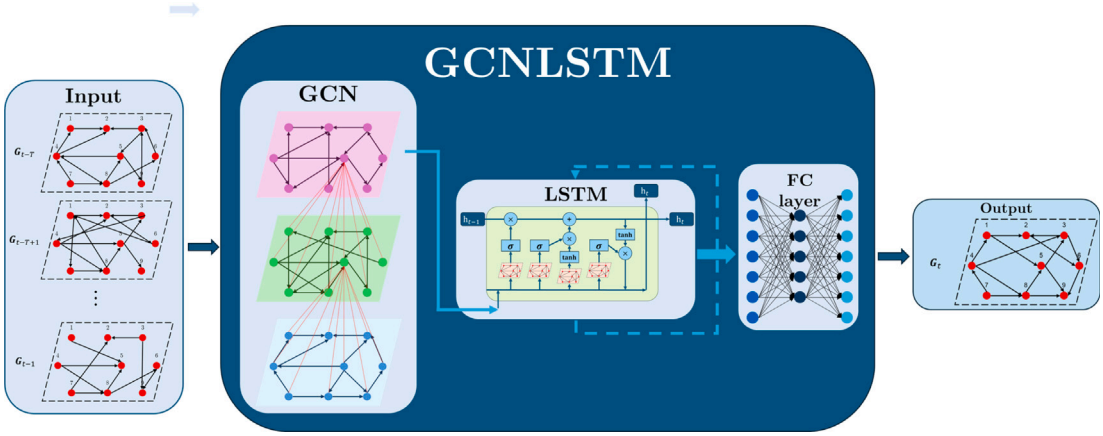


Fig. 14. An example schematic for combining GCNs and LSTMs for modelling spatial–temporal unstructured mesh data.

SE(3)-equivariant Convolutional Graph Network that is inherently invariant to translations and equivariant to rotations of the unstructured mesh, to accurately predict haemodynamic fields on high-resolution surfaces meshes of artery walls. Li et al. [228] trained convolutional and attentional GNNs on 3D Reynolds-averaged Navier-Stokes (RANS) data to accurately predict flow fields around wind turbines and power generation. Sanchez-Gonzalez et al. [229] developed graph network based simulators using message passing GNNs that can learn to simulate a wide range of challenging particle-based physical problems, including smooth particle hydrodynamics, rigid bodies and interacting deformable materials. By injecting strong inductive biases into the models, representing physical states with graphs of interacting particles and approximating physical dynamics by learned message-passing among nodes, the models were able to generalise to much larger graphs and longer time scales than those encountered during training. The work of [18] introduced MeshGraphNets which is a framework for learning mesh-based simulations of diverse physical problems, including aerodynamics, structural mechanics and cloth dynamics, all using message passing GNNs. In particular, the developed models are able to adaptively adjust the mesh discretisation during simulation and supports the learning of resolution-independent dynamics and scale to more complex discretisations at test time. Song et al. [230] proposed a GNN based mesh deformer for mesh movement-based mesh adaptation, which accelerates PDE solving. Barwey et al. [231] developed a multi-scale message-passing GNN autoencoder with learnable coarsening operations that can generate interpretable latent graphs that reveal regions important for flowfield reconstruction.

In fact, similar to CNNs, GNN layers are inherently scalable. However, challenges such as the exponential growth of neighbourhoods—commonly referred to as the neighbour explosion problem [232]—and the integration of GNNs with fully connected layers can hinder this scalability. These limitations reduce their effectiveness when dealing with heterogeneous datasets, such as simulation data on adaptive meshes. Multi-resolution GNNs for dynamical systems remain a very active research topic.

Compared to traditional CFD solvers, deep learning models, in particular GNNs, offer significant advantages for parallel and distributed computing. ML models are typically trained offline on fixed datasets, such as mesh snapshots, which decouples them from the complexities of dynamical mesh handling during simulation. Once the data (e.g., nodal values or graph representations) is preprocessed in GNNs, the irregularity of unstructured meshes has minimal impact on parallelisation. Moreover, modern ML frameworks (e.g., PyTorch, TensorFlow, JAX) are optimised for batched, tensor-based operations. Using techniques like mesh-to-graph transformations, even unstructured mesh data can be processed efficiently in uniform, GPU-parallel formats, reducing the communication overhead and load imbalance issues that commonly affect distributed CFD simulations.

3.3. Transformer and attention mechanism

3.3.1. Transformers: attention-mechanism

The advent of transformer models [21], originally developed within the domain of Natural language processing (NLP), has offered novel methodologies for addressing the challenges of sparse and irregular data. Characterised by their self-attention mechanisms, transformers have demonstrated remarkable ability to manage unstructured data and interpret data dependencies over long ranges, making them particularly suited for the intricacies involved in machine learning for computational physics.

An overview of the transformer model architecture is shown in Fig. 15. Note that for brevity, only the encoder part of the original transformer is depicted here. The decoder part has similar architecture and is designed for machine translation tasks, which are not relevant for many applications in computational physics. For readers interested in the decoder, please refer to the seminal paper on transformers [21]. Given an input sequence $\mathbf{x} \in \mathbb{R}^{n\times d}$, where n and d are the number of tokens and feature dimension of each token. In the context of language processing, an input sequence is a sentence and a token is the word embedding (i.e., encoded feature vector) of a word from the sentence. In computational physics, an input sequence can be a list consisting of all the nodes of an unstructured mesh and a token is the feature vector of a node. In the following part, we will introduce self-attention, the key component of transformers, and positional encoding mechanism in details.

Self-attention. The self-attention mechanism enables a model to selectively focus on different parts of a sequence when processing each token of that sequence. It achieves this by generating three vectors for each input token: a query vector (**Q**), a key vector (**K**), and a value vector (**V**), derived from the input through learned linear transformations (the MLPs shown in Fig. 15). The essence of self-attention lies in computing attention scores by taking the scaled dot product of a query vector with all key vectors, including itself, which are then used to create a weighted sum of value vectors. This process allows the model to weigh the significance of all other tokens when encoding each token, thereby encapsulating the context of each token within the sequence. To obtain vectors: query $\mathbf{Q} \in \mathbb{R}^{n\times d_q}$, key $\mathbf{K} \in \mathbb{R}^{n\times d_k}$ and value $\mathbf{V} \in \mathbb{R}^{n\times d_v}$, we define three learnable projection matrices $\mathbf{W}^Q \in \mathbb{R}^{d\times d_q}$, $\mathbf{W}^K \in \mathbb{R}^{d\times d_k}$, $\mathbf{W}^V \in \mathbb{R}^{d\times d_v}$, and apply them to the input as linear transformations:

$$\mathbf{Q} = \mathbf{x}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{x}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{x}\mathbf{W}^V. \tag{29}$$

Here we consider computing the simple scaled dot product attention. The attention scores (i.e., the measure of similarity between two vectors) are computed by taking the dot product of the query vector **Q** with all key vectors **K**, followed by a scaling factor to stabilise gradients during training. The scaling factor is usually the square root of the

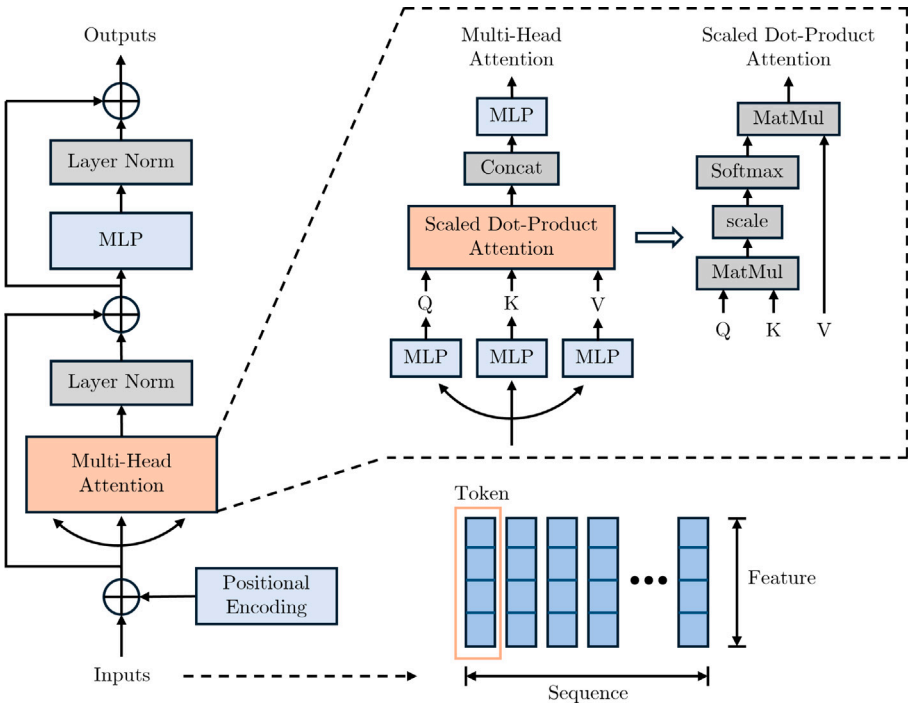


Fig. 15. Transformer architecture. For brevity, only the encoder part of the original transformer is shown as the decoder part has similar architecture.

dimension of the key vectors $\sqrt{d_k}$. To ensure that the attention scores sum up to 1 across the sequence for each query, a softmax function is applied after the attention score computation:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{30}$$

To enhance the ability to capture diverse feature and increase representation capacity, a multi-head attention mechanism [233] is usually applied instead of relying on a single head. When computing the multi-head attention, we project the input into queries, keys and values multiple times with different projection matrices $\mathbf{W}_i^Q \in \mathbb{R}^{d\times d_q}$, $\mathbf{W}_i^K \in \mathbb{R}^{d\times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d\times d_v}$. Given m heads, the attention of each head $\mathbf{h}_i$ is computed in parallel. All computed attentions $\{\mathbf{h}_i\}_i^m$ are concatenated together and projected with a projection matrix $\mathbf{W}^O \in \mathbb{R}^{md_v\times d}$:

$$\begin{aligned}&\text{MultiHeadAttention}(\mathbf{x}) = \text{Concat}(\mathbf{h}_1,\mathbf{h}_2,\dots,\mathbf{h}_m)\mathbf{W}^O,\\ &\text{where}\ \ \mathbf{h}_i = \text{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V\right),\end{aligned} \tag{31}$$

where m is the number of heads of MultiHeadAttention, d_v is the features dimension of the output after concatenating multi-head attentions. The computed attentions have the same size to value vectors V and each vector in the attention is a weighted summation of each vector in value vectors V. Intuitively, each vector in the attentions is an encoded vector containing aggregated local and global information from all other vectors. The amount of information aggregated from each vector to the encoded vector is determined by their similarity, i.e., weights or attention scores.

Positional encoding. Since the self-attention mechanism does not inherently process the sequential order of the input, transformers use positional encoding to incorporate information about the position of tokens in the sequence. This allows the model to understand the order of tokens, which is crucial for tasks in language processing.

In the original transformer paper [21], sine and cosine functions of different frequencies are proposed for positional encoding:

$$\mathbf{p}_{pos,2\iota} = \sin\left(\frac{pos}{10000^{2\iota/d}}\right), \tag{32a}$$

$$\mathbf{p}_{pos,2\iota+1} = \cos\left(\frac{pos}{10000^{2\iota/d}}\right), \tag{32b}$$

where the pos is the position of a token in a sequence and $\mathbf{p}_{pos,2\iota}$ is the 2ιth element of the d-dimensional token $\mathbf{p}_{pos}$ (i.e., $0 \le 2\iota \le d$). The positional encoding is added to input sequence $\mathbf{x}$ element-wisely for injecting position information explicitly for each token. Therefore, the query, key and value with positional encoding are extended as:

$$\mathbf{Q} = (\mathbf{x}+\mathbf{p})\mathbf{W}^Q, \quad \mathbf{K} = (\mathbf{x}+\mathbf{p})\mathbf{W}^K, \quad \mathbf{V} = (\mathbf{x}+\mathbf{p})\mathbf{W}^V. \tag{33}$$

The positional encoding introduced above is designed for language processing. There are different positional encoding methods tailored for graph data structures, e.g., [234,235]. For more discussions about positional encoding methods, which are out of scope of this review, readers can refer to [236] for more details.

3.3.2. Applications in computational physics

Although transformers were originally proposed for language processing tasks, their strong modelling abilities and capacities have drawn increasing research interest in machine learning for areas among computational physics, for examples, Hamiltonian dynamics [237], particle physics [238], computational fluid dynamics [22,239], weather and climate forecasting [240]. Transformers can be combined with GNN to capture long-term temporal dependencies through its attention mechanism for long sequence prediction in physical simulations [241], or capturing long-distance spatial information for mesh adaptation [242]. The self-attention mechanism underpins the transformers' ability to aggregate long-range (or global) information and naturally to handle unstructured data. The explicit positional encoding enables transformers to understand and incorporate the order or position of tokens (or nodes in unstructured meshes). These features present advantages comparing to CNNs and GNNs in handling spatial data. A visualisation of the comparison between different models on processing unstructured mesh data is shown in Fig. 16. In addition to using transformers as pure data-driven model for physical system modelling, there is a special interest in building neural PDE solvers using transformers especially in neural operator learning [243]. Intuitively, the self-attention mechanism can be interpreted as a learnable Galerkin projection [244] or a learnable kernel integral [245,246] in neural operator learning methods. To handle both uniform and non-uniform discretisation grids, OFormer [234] is proposed based on an encoder–decoder architecture with self-attention

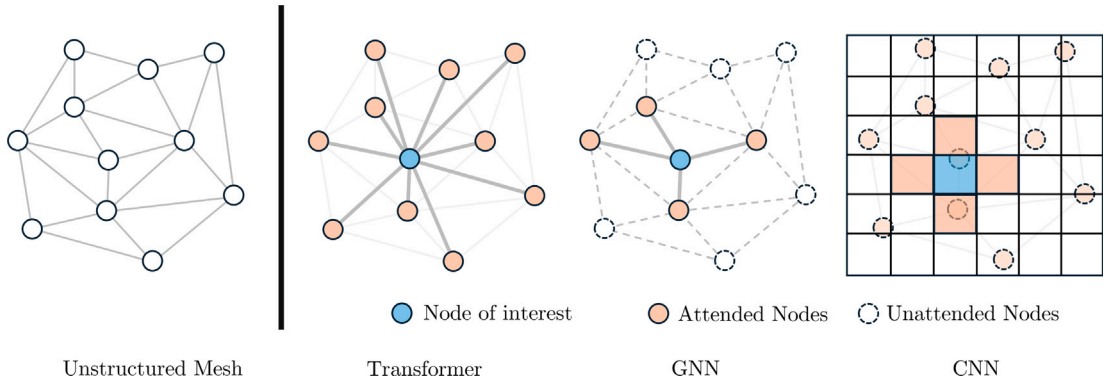


Fig. 16. Comparison between models for the processing of unstructured mesh data. A transformer model employs a self-attention mechanism to consider all mesh nodes. Single layer GNNs attend their direct neighbours. CNNs apply a fixed size kernel on a structure grid with interpolated data from the original unstructured mesh.

and cross-attention. The General Neural Operator Transformer [247] proposes a heterogeneous normalised attention layer to flexibly handle unstructured meshes, multiple and multi-scale input functions. The quadratic-complexity of standard scaled-dot product attention computation limits its applications on large scale problems. To improve the efficiency and scalability, linear attention is investigated for PDE modelling [244]. FactFormer [248], on the other hand, proposes a computationally efficient low-rank surrogate for the full attention based on an axial factorised kernel integral. Large-scale pre-training has emerged as a potential approach demonstrating robust generalisation capabilities across a range of downstream tasks in both natural language processing [249] and computer vision [250] domains. There have been initial attempts to explore pre-training for PDEs [251–254], in which transformers serve as backbones.

There has also been significant work applying transformers to 3D irregular meshes, addressing tasks such as mesh recovery and 3D mesh generation [255–260]. A key aspect of these approaches is the use of spatial attention mechanisms, which allow the models to effectively capture and utilise the spatial relationships within the mesh data. For instance, in the context of mesh recovery, spatial attention enables the transformers to focus on crucial areas of the mesh, leading to more accurate reconstructions. The Deformable Mesh Transformer, as explored in recent work [258], leverages these attention mechanisms to improve the accuracy of human mesh recovery by dynamically adjusting the attention across different mesh regions, effectively handling complex deformations. Additionally, transformer-based models have shown promise in generating 3D meshes by focusing on the intricate spatial dependencies within the data [256], which is crucial for creating detailed and coherent structures. These advances highlight the growing role of spatial attention in enhancing the performance of transformers for 3D mesh tasks.

In summary, transformers have shown promising abilities in modelling complex physical systems of computational physics. Compared to CNNs and GNNs, transformers can naturally handle unstructured meshes as well as capture long-range dependencies using the self-attention mechanism. Furthermore, leveraging their scalability and parallelisation capabilities, transformers are extensively utilised in large-scale pre-training models. They serve as foundational backbones for constructing models that facilitate the learning of PDEs in the computational physics area.

3.4. Summary and comparison

It is clear that the three families of approaches introduced in this section have been widely adopted in the computational science community to handle spatially unstructured and irregular data. However, each of them may exhibit certain advantages and disadvantages depending on the application field. Therefore, we believe a qualitative comparison would be beneficial to highlight the strengths of each family of approaches.

CNNs capture local patterns efficiently by applying the same filter across different parts of the input, leveraging translational invariance. Conventional CNNs struggle with irregular data such as unstructured meshes, requiring interpolation onto regular grids, which introduces potential errors. An alternative solution could be 1D CNNs, particularly with mesh-dependent reordering as introduced in Section 3.1.2. Although 1D CNNs can seamlessly handle unstructured grid data, their unidimensionality constrains their ability to capture relative and global position awareness within the neural network. Standard CNNs are scalable, meaning they can be trained and evaluated on data of varying resolutions automatically. However, this review specifically focuses on their application to unstructured grid data, where preprocessing or interpolation is often required, reducing their overall scalability.

GNNs are designed to work with graph-structured data, where the concept of position is not as straightforward as in structured grid-like data structures. Instead of relying on positional information, GNNs focus on the structure and relationships between nodes, making them inherently scalable to data of varying dimensions. They are aware of the node's position in terms of its connectivity and neighbourhood in the graph. Therefore, GNNs excel at capturing the relational invariance and dependencies between elements, but struggle to understand the absolute or relative positions which are important in computational physics. In addition, as GNNs deepen, they can suffer from over-smoothing [261], where node features become too homogenised, reducing model performance in tasks requiring fine distinctions.

Transformers are the most explicitly position-aware, designed to incorporate position information directly into their processing, which makes them highly effective for a wide range of tasks that require an understanding of element order or position within the data. An important strength of the attention mechanism is position awareness, i.e., the ability to understand and incorporate the order or position of elements. In the context of machine learning in computational physics, the "position" indicates both the relative and absolute spatial positions of elements in data structures such as meshes. On the other hand, transformers, as the foundation of many 'large' AI models, require considerably more data to train compared to CNNs and GNNs, a phenomenon often referred to as 'data-hungry' [262].

A comparison summary of the abilities of single-layer CNNs, GNNs and transformers is shown in Table 1. The column 2DCNN* indicates 2D CNN methods with grid reordering, interpolation or transformation. A growing body of research works has explored the comparative strengths of CNNs, GNNs and Transformers for solving specific PDEs and fluid dynamics tasks on unstructured meshes, seeking a balance between simplicity, adaptability, and computational trade-offs. CNNs, such as U-Nets, have demonstrated strong accuracy and efficiency on

Table 1
Summary of neural network and linear projection methods (✓: correct; ×: incorrect; ○: partially correct).

Methods	POD/DMD	MLP	1DCNN	2DCNN*	GNN	Transformer	PINNs
Global information	✓	✓	×	×	×	✓	×
Grid information	×	×	×	×	✓	✓	○
Adaptive mesh	×	×	○	○	✓	✓	✓
Computational efficiency	✓	×	○	○	×	×	×
Physics information	×	×	×	×	×	×	✓
Position awareness	×	×	○	✓	×	✓	✓
Dimension scalability	×	×	○	○	✓	✓	✓

structured and even unstructured meshes, as shown by [263], challenging the assumption that GNNs are always necessary. However, studies such as [221,264] show that GNNs equipped with advanced architectures (e.g., Gaussian mixture convolutions or Flow Completion Networks) significantly outperform CNNs in capturing spatial dependencies and managing incomplete or irregular data in 3D CFD scenarios. Building on these insights, recent research has emphasised the advantages of incorporating transformer models. The recent work of [265] proposed a hybrid architecture that combines GNNs' local spatial modelling with transformers' temporal attention, achieving efficient and accurate predictions for unsteady flow on unstructured meshes. Similarly, [266] introduced the Mesh Transformer, replacing iterative GNNs updates with global attention for improved long-range interaction modelling. This trend is further extended in [267] with a multiscale GNNs framework that integrates Transformer-based message passing and Adaptive Mesh Refinement to enhance multiphysics simulation accuracy. In parallel, [268,269] highlighted the limitations of CNNs in modelling long-range dependencies and fine-scale structures, showing that transformer-based models, particularly those using hierarchical attention like Swin-Transformers, achieve superior performance in fluid turbulence reconstruction and flow compression tasks. Together, these studies underscore that while CNNs remain competitive in structured contexts, GNNs and transformer-based models offer significantly improved flexibility, accuracy, and scalability for learning on unstructured mesh data.

Regarding the accuracy of predictive or reconstruction models based on these neural network architectures, it is crucial to ensure that the models maintain strong performance when applied to unseen scenarios, across varying geometries, boundary conditions, and physical regimes. While GNNs and meshless neural networks such as standard PINNs (see Section 4.1) are often promoted as flexible and physically grounded frameworks, their robustness in extrapolative settings remains an open challenge. For example, [270] showed that message-passing GNNs trained on laminar flows over a fixed set of shapes struggled to generalise to novel geometries with sharper features or different Reynolds numbers. Similarly, [271] conducted a systematic evaluation of PINNs and found that they fail to generalise reliably when applied to domains with unseen boundary conditions or discontinuous solutions, often requiring problem-specific tuning or additional loss weighting strategies. The work of [272] also reported limited transferability of PINNs when modelling wave propagation across domains with heterogeneous media, despite incorporating governing equations. These findings suggest that despite their theoretical appeal, both GNNs and PINNs still face critical limitations in their generalisability, particularly in real-world applications involving complex, unseen geometries.

While the architectural flexibility and performance of CNNs, GNNs, and transformers have been widely explored, a critical aspect that remains under-addressed is uncertainty quantification (UQ) and the choice of evaluation metrics in the context of unstructured grids. Traditional metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) are designed for structured domains and fail to capture topological nuances and local resolution variations inherent in unstructured meshes. Similarly, many existing UQ techniques assume uniformly sampled data, making them ill-suited for irregular, sparse domains where mesh quality varies spatially. Recent studies [266,267,269] emphasise the need for mesh-aware metrics, such as graph-based Wasserstein distances or geometry-weighted errors, that reflect the physical relevance and numerical fidelity of the predictions. Furthermore, reliable UQ in such domains often requires incorporating spatially adaptive uncertainty representations conditioned on mesh connectivity or local flow characteristics. These challenges underscore the importance of developing new, domain-specific evaluation frameworks for robust assessment of ML methods on unstructured data. For more details about uncertainty quantification in dynamical systems, interested readers are referred to recent review papers [82,273].

4. Learning paradigms with unstructured data

Machine learning performance on unstructured data is influenced not only by neural network structures but also by training objectives, including loss function design and data augmentation strategies. This section reviews Physics-Informed Neural Networks with meshless loss functions, reinforcement learning for mesh generation, and generative AI approaches for handling unstructured grid data. These workflows are largely independent of the neural network structure, ensuring adaptability to those introduced in Section 3.

4.1. PINNs: a meshless solution

PINNs mark a significant evolution in scientific machine learning, by embedding the physical laws described by the underlying differential equations in the neural network [23]. Traditional numerical methods, while foundational in engineering, struggle with challenges such as mesh dependency and computational burdens in high dimensions. Early neural network applications were limited to purely data-driven techniques but recent advances in automatic differentiation have revitalised this approach, with PINNs at the forefront, offering solutions to forward and inverse problems by enforcing physical laws within the learning process [274,275].

PINNs excel in parametrising high-dimensional PDEs, incorporating parameters directly into the training data to navigate expansive parameter spaces efficiently, which is a significant advantage over conventional numerical techniques [276]. This capability is further enhanced by Physics-Informed Deep Operator Networks, which learn an operator learning architecture to solve parametric problems [277].

Fig. 17 shows the schematic of a parametric PINN for a 3D time-dependent problem. The inputs to the neural network are the spatio-temporal coordinates x, y, z, t along with a set of parameters λ. Although one can obtain these spatial coordinates from the nodal points of an unstructured mesh, employing sophisticated pseudo-random sampling techniques, such as Latin hypercube sampling, Sobol sequences [278], Halton sequences [279], or Hammersley sequences [280], offers a more efficient alternative. These techniques generate unstructured point clouds that require significantly fewer points for effective domain sampling, compared to using the nodal points from unstructured meshes directly.

PINNs inherently manage unstructured data more efficiently than conventional architectures like CNNs and RNNs due to their ability to integrate differential equations directly into the learning process. CNNs, primarily designed for structured grid-like data, such as images, often struggle with irregular, non-gridded data formats without extensive pre-processing to regularise the input space. RNNs, while

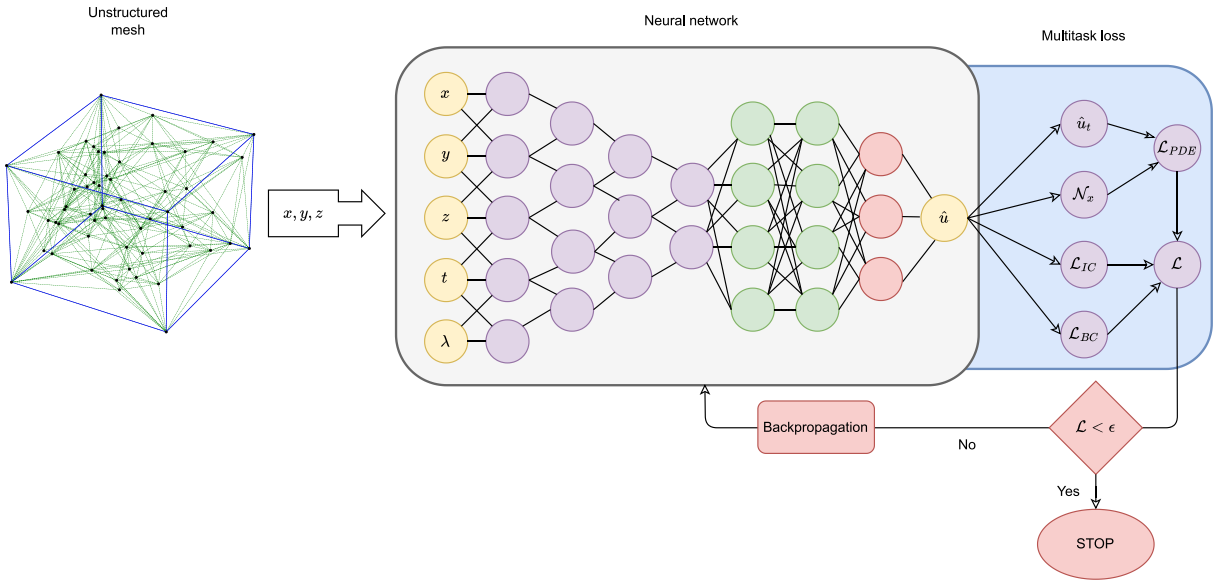


Fig. 17. Schematic of a parametric PINN for a 3D time dependent problem.

adept at handling sequences, similarly face challenges with spatial irregularities and multidimensional data. In contrast, PINNs leverage the underlying physical laws, represented through differential equations, to learn the solution to the PDE in the domain. This capacity allows PINNs to learn complex patterns with fewer data points [281–283]. The performance of PINNs and recently developed KANs (Kolmogorov–Arnold Networks) [284] on irregular geometries have been extensively examined in the recent works of [285,286].

The output from the PINN is the solution to the PDE. This output is constrained to satisfy a multitask loss consisting of the initial condition, boundary conditions and the PDE. These loss terms are often evaluated via a Mean squared error (MSE).

In the analysis of incompressible fluid dynamics, it is fundamental to enforce the conservation of mass, expressed as zero divergence of the velocity field. For bounded domains with solid boundaries, such as in pipes or channels, the no-slip condition is often imposed. The mathematical formulation is as follows:

$$\nabla \cdot \mathbf{x} = 0, \tag{34a}$$

$$\mathbf{x} = \mathbf{0} \text{ on } \partial\Omega. \tag{34b}$$

Here, $\mathbf{x}$ represents the velocity field vector, comprising components u, v and w in the three directions respectively. $\partial\Omega$ denotes the boundary of the domain. The PINN architecture for divergence-free flow problems incorporates two primary loss components to guide the training process: the PDE loss ($\mathcal{L}_{PDE}$) and the Boundary Condition loss ($\mathcal{L}_{BC}$). The individual loss terms can be formulated as:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_p}\sum_{k=1}^{N_p} \left\|\nabla \cdot \hat{\mathbf{x}}(\mathbf{i}_k)\right\|^2, \quad \mathcal{L}_{\text{BC}} = \frac{1}{N_b}\sum_{k=1}^{N_b} \left\|\hat{\mathbf{x}}(\mathbf{i}_k) - \mathbf{0}\right\|^2, \tag{35}$$

where, $\hat{\mathbf{x}}$ represents the PINN-predicted solution. The vector $\mathbf{i}_k$ denotes the spatial coordinates of the kth point within the domain or on the boundary, respectively for PDE and boundary condition losses. N_p is the total number of collocation points used to evaluate the PDE loss, distributed throughout the domain Ω. N_b is the total number of collocation points on the boundary $\partial\Omega$, used to evaluate the boundary condition loss. Additionally, an optional data loss term ($\mathcal{L}_{\text{Data}}$) can be included if the solution to the PDE is known at sparse locations:

$$\mathcal{L}_{\text{Data}} = \frac{1}{N_d}\sum_{k=1}^{N_d} \left\|\hat{\mathbf{x}}(\mathbf{i}_k) - \mathbf{x}(\mathbf{i}_k)\right\|^2, \tag{36}$$

where N_d represents the number of data points where direct measurements of the velocity field $\mathbf{x}(\mathbf{i}_k)$ are available. For benchmarking the efficacy of PINN-based frameworks, Mean absolute error (MAE) is calculated between the PINN predicted solution and a known solution (analytical or numerical), though PINNs themselves are employed as solvers for PDEs:

$$\mathcal{L}_{\text{MAE}}(\hat{\mathbf{x}}) = \frac{1}{N_v}\sum_{k=1}^{N_v} \left\|\hat{\mathbf{x}}(\mathbf{i}_k) - \mathbf{x}(\mathbf{i}_k)\right\|, \tag{37}$$

where $\mathbf{x}(\mathbf{i}_k)$ represents the true velocity field at the point $\mathbf{i}_k$, and N_v is the number of data points used for validation. This metric $\mathcal{L}_{\text{MAE}}$, which is not included in the training loss function, provides a measure of the average magnitude of error across all sampled points.

Importance sampling is a technique used to enhance the efficiency of the training process in machine learning models, including PINNs. A probability distribution is first constructed that is proportional to the loss distribution observed across the computational domain. This distribution then guides the sampling process, whereby training points are more frequently selected from regions where the current model exhibits higher training loss. This method ensures that the neural network training concentrates on areas of the domain where the model is under-performing, thereby enhancing overall model accuracy. Essentially, this strategy allows for an adaptive allocation of computational resources, focusing efforts on 'problematic' areas much like adaptive mesh refinement in traditional computational methods [287–289].

Addressing the disparity in loss term magnitudes, which could bias learning towards boundary condition at the expense of accurately solving differential equation, balancing coefficients have been introduced. This, along with advancements in adaptive coefficient adjustment, underscores efforts to equalise the contribution of each loss component to the learning process [290–292].

The development of adaptive activation functions represents another stride, introducing trainable parameters that fine-tune the activation function's slope, thus enhancing the network's ability to model complex patterns and behaviours [293,294]. Several neural network architectures have been developed to project the low-dimensional input of the PINN to a higher dimension. This strategy is useful in addressing pathologies such as the spectral bias in neural networks, particularly beneficial for learning high-frequency functions indicative of abrupt changes such as discontinuity in the solution [295–298].

Domain decomposition within PINNs, analogous to finite element methods but with enhanced flexibility, enables tackling complex geometries and discontinuities. Techniques like Conservative Physics-Informed Neural Network, Extended Physics-Informed Neural Network

and parallel PINNs have emerged, focusing on computational efficiency and modelling for discontinuities [299–301]. The ability to decompose the domain has been further enhanced in Finite Basis Physics-Informed Neural Network [302] by employing overlapping subdomains. Additionally, the Geometry Aware Physics-Informed Neural Network [303] effectively compresses the complexity of irregular geometries, typically represented through unstructured meshes, into a latent space. This latent representation, alongside spatial coordinates x, y, z, serves as input for a subsequent network within the PINN framework. A separate neural network is employed specifically to enforce boundary conditions with hard constraints during the training phase of the PINN.

In recent developments, input encoding strategies for PINNs have been crucial in handling unstructured data more effectively. These encodings can generally be categorised as Fourier type, involving combinations of sine and cosine functions, or non-Fourier types such as radial basis function and hash encoding. In [304], Fourier features were utilised to encode spatial coordinates of unstructured data points, alongside implementing a hard-constrained output in the neural network, which effectively eliminates the boundary condition loss ($\mathcal{L}_{BC}$). This study also introduced an active sampling technique, an advancement over traditional importance sampling, to efficiently resample unstructured points. Conversely, Zeng et al. [305] employed non-Fourier radial basis function encoding to manage inputs from unstructured datasets, complemented by a custom finite difference scheme designed to compute derivatives in scenarios involving discontinuous solutions. Meanwhile, Huang et al. [306] demonstrated that hash encoding could significantly accelerate PINN training, achieving up to a tenfold decrease in the training time. These encoding strategies not only enhance the computational performance of PINNs but also significantly improve the management of unstructured data, influencing the convergence behaviours and learning dynamics. This necessitates careful consideration of their integration into existing architectures, particularly as the discontinuous nature of hash encoding poses unique challenges in derivative handling, impacting the stability and robustness of the model.

Additionally, a significant advancement in addressing the limitations of conventional PINNs, particularly their struggle with temporal dependencies in dynamic physical systems, has been the development of a transformer-based framework known as PINNsFormer [307]. This framework utilises multi-head attention mechanisms not only to more accurately capture temporal dependencies but also to efficiently process unstructured data by transforming point-wise inputs into pseudo sequences. This adaptation enhances the model's ability to deal with non-uniform and scattered data typically encountered in complex physical scenarios.

Recent advancements in operator learning frameworks like DeepONet [277] and Fourier neural operator (FNO) [203] have extended their application to unstructured grid data, addressing a long-standing limitation in solving PDEs across irregular geometries. For instance, the Mesh-Independent Neural Operator [308,309] and Geo-FNO [310] have enabled efficient transformations of non-uniform domains into latent spaces, allowing for operations like fast Fourier transform to function seamlessly on arbitrary geometries. Similarly, frameworks such as Non-Uniform Neural Operator [311] leverage domain decomposition techniques like K-D trees to handle non-uniform data while maintaining computational efficiency. FNO with localised integral and differential kernels further address challenges of over-smoothing by introducing locally supported kernels, effectively bridging the gap between global and local feature representations [312].

DeepONet variants have also seen remarkable progress in addressing problems related to unstructured grids. Geom-DeepONet [313] augments inputs with signed distance functions and employs sinusoidal representation networks to predict field solutions across parametrised 3D geometries, demonstrating robust generalisation to unseen configurations. Physics-Constrained DeepONet [314] integrates physical laws like divergence-free constraints to enhance learning efficiency, especially with sparse data. Furthermore, Enriched-DeepONet [315] and Decoder-DeepONet [316] tackle challenges such as moving-solution operators and unaligned observation data, improving accuracy by orders of magnitude compared to conventional approaches. These advancements underscore the growing capability of neural operators to efficiently solve PDEs on unstructured grids.

In summary, PINNs and neural operators represent a significant advancement in computational science, combining the strengths of neural networks with accurate physical modelling to address many limitations of traditional numerical methods for solving differential equations. By integrating input encoding techniques such as Fourier and non-Fourier methods, and employing novel strategies like multi-head attention mechanisms from transformer-based models, PINNs are uniquely equipped to handle unstructured data. Additionally, new techniques in sampling, balancing loss terms, and adapting activation functions, along with domain-decomposition, enable PINNs to work effectively with experimental data and solve complex physical equations.

4.2. Reinforcement learning for unstructured mesh generation

4.2.1. Introduction to reinforcement learning

Reinforcement learning (RL) is a machine-learning technique where an agent learns to make decisions by interacting with its environment [317]. The learning process is guided by the rewards or penalties received from trial-and-error interactions, resulting in an optimal decision-making policy that maximises cumulative rewards over time. Since a complex problem could often be divided into a sequence of manageable sub-tasks, the overall performance hinges on the ability to effectively solve these sub-problems.

RL has recently been applied to solve unstructured mesh generation problems, due to its underlying mechanism of recursive interactions between element extractions and its domain boundary environment [318–320]. The work of [321,322] formulated the mesh generation problem in terms of sequential decision making problems and developed RL-based methods to generate quadrilateral meshes for arbitrary two-dimensional (2D) geometries. The overall mesh generation procedure is shown in Fig. 18. The meshing process is formulated as a Markov decision process, consisting of a set of boundary environment states $\mathcal{B}$, a set of possible actions $\mathcal{X}(b)$, a set of rewards $\mathcal{R}$, and a state transition probability

$$P(B^{t+1} = b', R^{t+1} = r | B^t = b, X^t = x), \tag{38}$$

where b', $b \in \mathcal{B}$, $x \in \mathcal{X}(b)$, $r \in \mathcal{R}$, b' is the new state at $t+1$ and r is the reward after action x on the boundary state b. It is important to note that the state (solution) variable x in this section represents the action of mesh generation, unlike in other sections where it denotes the provision of simulation or prediction results.

The mesh generator at each time step t, acting as the RL agent, observes a state B^t from the environment, which is the contour of the geometry consisting of a set of vertices and segments. It then performs an action X^t to generate a new element from the given state based on an implicit or explicit policy. The environment responds to the action by excluding the newly generated element and transits into a new state B^{t+1}, which is the updated geometry contour. It provides a reward or penalty R^t simultaneously that measures the quality of the generated element and the updated contour, such as a value of 1 for a well-balanced quality and -1 for an imbalance. The latter could potentially lead to sharp angles and narrow regions. This element generation process continues until the updated contour includes only four vertices, forming the last element automatically. Eventually, it produces a sequence of $[B^0, X^0, B^1, R^1, X^1, \ldots]$. The agent's goal is to learn such a policy $\pi(x|b)$ that maximises the cumulative rewards G_t^r,

$$G_t^r = \sum_{k=t+1}^{T} \gamma^{(k-t-1)} R_k, \tag{39}$$

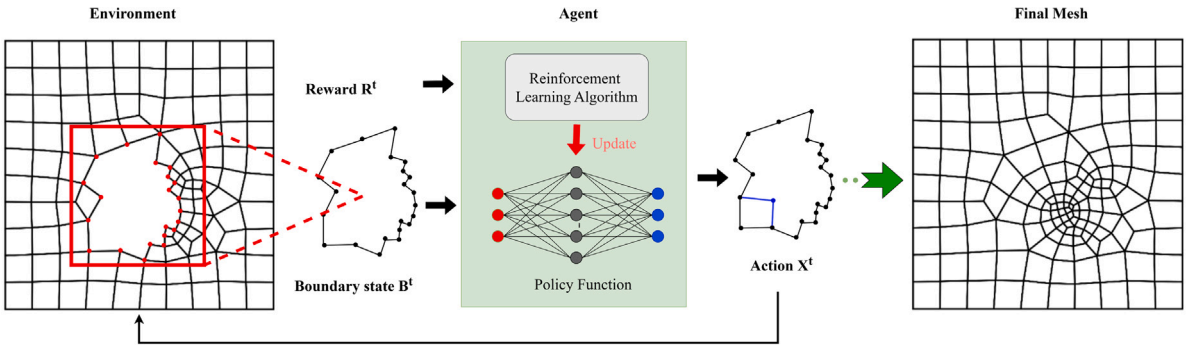


Fig. 18. Reinforcement learning for unstructured mesh generation.

where $0 \leq \gamma \leq 1$ is a discount rate that determines the weight between short-term and long-term rewards, and T is the total time step.

In deep RL, the policy π_{Φ} is represented by a nonlinear neural network with parameters θ, which were updated during the training process. Different types of deep RL algorithms could be applied to find an optimal policy π_{θ}^{*} that maximises an objective function J defined as the expectation of the return. π_{θ} is updated by $\nabla_{\theta} J$ which is calculated using the deterministic policy gradient algorithm by applying the chain rule to J as follows:

$$\nabla_{\theta}(J) = \mathbb{E}_{(b^t, x^t)}\left[\nabla_{\theta} Q^{v}(b^t, x^t)\right], \tag{40}$$

where $Q^{v}(b^t, x^t)$ is an action value function defined by

$$Q^{v}(b^t, x^t) = \mathbb{E}_{\pi}\left[R^t | b^t, x^t\right]. \tag{41}$$

Here the notation Q^{v} for the value function is chosen to avoid conflict with the query vector notation of the transformers. The action value function estimates the long-term benefit of taking an action for a state, which is how well the newly generated element will contribute to the optimal mesh quality. The mesh quality is measured by metrics related to the mesh's geometrical and topological properties [323]. Common geometrical metrics for a mesh include minimum and maximum angles, edge ratio, aspect ratio, stretch, taper, skewness, and scaled Jacobian, whereas topological metric includes singularity. Generally, a mesh cannot achieve high scores on every metric because of the complexity of geometries and specific computational requirements. The choice of metric is often related to downstream applications. The stepwise reward in the Q-function needs to balance the element quality and mesh quality because the optimal individual element does not always lead to optimal meshes. Pan et al. [321] demonstrated that a reward function considering the trade-off between element quality and the remaining contour could achieve overall good performance.

Once the reward function is defined, different types of RL algorithms could be applied to estimate the Q-function and the optimal policy. Pan et al. [321] applied Advantage Actor–Critic to learn an initial meshing policy and then trained a feedforward neural network as the final policy with high-quality samples from the RL agent. With the similar formulation, Tong et al. [324] further expanded the action space of extracting a new element. To achieve a stable learning efficiency, Pan et al. [322] solely implemented Soft Actor-Critic to learn the meshing policy, which has enabled it for more complex geometries.

4.2.2. Reinforcement learning for mesh optimisation

Mesh optimisation for complex systems plays an important role in dynamically refining mesh regions with low or high solution variability, facilitating a favourable trade-off between computational speed and simulation accuracy. The application of RL to this research area has recently started. Wang et al. [325] proposed a smoothing method to improve the quality of triangular meshes by combining a heuristic Laplacian method with the Deep Deterministic Policy Gradient algorithm. The learned policy maximises the overall mesh skewness quality by adjusting the positions of free nodes.

Some research directly optimise meshes towards an accurate and efficient numerical approximation to solve PDE. Reinforcement learning is applied to automate this process, sidestepping a large amount of heuristic rules. The recent work of [326] formulated adaptive mesh refinement into a Markov decision process with variable sizes of state and action spaces. It makes elementwise decisions with policy trained by REINFORCE and Proximal Policy Optimisation (PPO), to minimise the final PDE solution error. Gillette et al. [327] applied Proximal Policy Optimisation to train a policy that marks a set of elements to be refined based on existing error estimates. The specific refinement strategy was based on prior knowledge. Instead of only focusing on refining meshes, Foucart et al. [29] formulated the Adaptive Mesh Refinement as a partially observable Markov decision process and learned a policy that could de-refine mesh elements with local surrounding information. Lorsung et al. [30] applied a deep Q network to iteratively coarsen meshes, thereby reducing simulation complexity while preserving the accuracy of the target properties.

To avoid iteratively refining the elements, some studies applied multi-agent RL. Yang et al. [328] further investigates multi-agent GNN with a team reward to boost the refinement speed for arbitrary meshes. Freymuth et al. [329] formulated adaptive mesh refinement as a Swarm Markov Decision Process, treating each mesh element as an agent and training all agents under a shared policy using GNNs. Dzantic et al. [330] proposed a multi-agent PPO to learn anticipatory refinement strategies, specifically for discontinuous Galerkin finite difference method.

A few studies have used RL to optimise the shape of geometries with existing mesh generators, such as fin-shaped geometries for optimal heat exchange [331] and blade passages with optimal meshing parameters [332]. Some complex geometries often require a well decomposition before meshing. Diprete et al. [333] used PPO to train an agent to perform optimal cuts on Computer Aided Design models, decomposing them into well-shaped rectangular blocks suitable for generating high-quality meshes.

In summary, RL is starting to make significant strides in generating and optimising unstructured meshes for various engineering applications, enhancing both computational efficiency and simulation accuracy. RL generally reduces reliance on heuristic rules and enhances the accuracy of numerical approximations for PDEs. RL-based unstructured meshes could be applied to fields, including fluid dynamics by capturing complex flow patterns, refining meshes in structural engineering by predicting stress distributions and material behaviour, and modelling biological tissues and organs, which facilitates better surgical planning and medical device design. Future research in RL for unstructured meshes includes:

1. Leveraging multi-agent RL to coordinate meshing and refinement strategies across mesh elements, as seen in the use of GNNs and swarm intelligence.
2. Extending RL applications to optimise geometry shapes in conjunction with mesh generation, improving design efficiency in fields like heat exchange and fluid dynamics.

Fig. 19. The trilemma of the four most popular generative model types: all models can be framed into this triangle, and there is yet to find a method that could concurrently exhibit the three assets.
Source: Inspired by Figure 1 from Xiao et al. [350].

3. Developing RL agents capable of performing optimal preprocessing on Computer Aided Design models, enabling better decomposition for high-quality mesh generation.
4. Evaluating the RL-based methods in diverse applications, including fluid dynamics, structural engineering, and biomedical engineering.

4.3. Generative AI models with unstructured grid data

4.3.1. Introduction to generative models

Similar to physics-informed machine learning (Section 4.1) and reinforcement learning (Section 4.2), generative modelling characterises a set of techniques towards a specific inference goal, independent from specific neural network architectures. As such, all previously discussed architectures for handling unstructured data can also be applied to generative modelling. Generative modelling seeks to learn an efficient sampler from a state distribution $\mathbf{x} \sim p_\theta(\mathbf{x})$ as parametrised by the weights and biases θ of a neural network. By modelling a conditional distribution $\mathbf{x} \sim p_\theta(\mathbf{x} \mid \mathbf{c})$, where $\mathbf{c} \in C$ is the conditioning information, e.g., initial conditions, a coarse-grained field, or observations, drawn from the conditioning space C, such a sampler can solve many tasks like forecasting, downscaling, or Bayesian inference. To avoid specifying an explicit distribution, implicit generative modelling represents the distribution through samples generated with a learnable generator applied to samples from a known and possibly simpler distribution [334]. For a more detailed overview of generative modelling, we refer to the books of Tomczak [335] and Murphy [336].

A straightforward way to learn such a generative model is to maximise the data likelihood given the assumed model, known as the maximum likelihood approach. Most models for generative modelling can be trained this way: energy-based models [337,338] and normalising flows [339,340] are examples, as well as Autoregressive models (ARMs) [341,342], which often rely on maximum likelihood estimation. Aiming to maximise a lower bound on the data likelihood, VAEs [343,344] and Denoising diffusion models (DDMs) [345–347] optimise the so-called Evidence lower bound (ELBO). Differing from that maximum likelihood approach, GANs [348,349] are trained with an adversarial loss: concurrently to the generator, a deep learning-based discriminator, or critic, is trained to discriminate between true data samples and samples from the generator.

GANs can generate high-quality samples with high throughput, but are notoriously unstable to train [351] and result in mode collapse, where only partial modes of the data are generated. Contrastingly, maximum likelihood approaches are more stable and easier to train, and they can exhibit much higher sample diversity than GANs. VAEs can generate samples in one step like GANs, but they often produce lower quality samples due to data compression. Conversely, DDMs and ARMs can generate high-quality samples but rely on slower multi-step sampling. Thus, there is a trilemma between training stability, sample quality, and generation speed [350]. Fig. 19 categorises the main generative modelling approaches within this trilemma, while a general overview over the most popular approaches and their original citations is shown in Table 2. Since DDMs are currently the most popular generative modelling approach, we concentrate on its application to unstructured data; however, most of the reviewed applications could also work with other types of generative modelling.

DDMs establish bidirectional pathways between the distribution of training data and a stationary noise distribution, typically a Gaussian distribution [345,346]. In the forward path, noise gradually replaces information in the data through a diffusion process. Conversely, the backward path requires training neural networks to denoise the data given a noised field and conditional information. Trained across all noise magnitudes, the neural network can then iteratively map initial fields from the stationary noise distribution back to data space. Being inherently probabilistic, the diffusion process is governed by a Stochastic differential equation (SDE) [34], and the backward path corresponds to a reverse SDE [352]. This connects DDMs to score-based generative modelling [347,353,354], where the neural network approximates the data distribution's score. Additionally, this connection allows the drawing of similarities with more traditional physical modelling approaches [355–357]. There exists also a deterministic Ordinary differential equation (ODE) for the backward path [34,358], yielding the same marginal distribution as the reverse SDE. DDMs enable advanced generative tasks such as text-to-image or text-to-video generation, enhancing generative training stability and quality [359]. However, their iterative nature incurs high computational costs and ongoing research aims to mitigate these costs [360].

4.3.2. Generative models applied on unstructured data

Early application of generative models to unstructured data are especially based on GANs or VAEs. Specific examples are their application to material modelling [361] and flood forecasting [362]. Combining these methods, Quilodrán-Casas and Arcucci [363] uses together a VAE, a GAN, and a RNN in latent space for forecasting on unstructured grids, as exemplarily shown in Fig. 20. Such applications of GANs on spatio-temporal problems are further surveyed in Gao et al. [364].

However, the first applications of DDMs highlighted their potential for complex fluid prediction [39,365], including for unstructured grids. In Google DeepMind's GenCast [38], a DDM designed for global weather prediction outperforms all other ensemble-based prediction methods, including ensemble forecasts from the European Centre for Medium-range Weather Forecasting (ECMWF). GenCast uses a graph neural network inspired by the GraphCast model [366], incorporating graph transformer blocks to represent data on a spherical grid, an approach also applicable to unstructured grids.

DDMs based purely on transformer blocks and attention mechanisms [367] are well-suited for unstructured data. Additionally, the latent mapping in latent diffusion models [360,368] can be readily adapted using the methods from Section 3.1.2 to transition from an unstructured to a latent structured grid, facilitating the application of diffusion models. However, as discussed in [38], the diffusion process must isotropically replace information with noise, potentially requiring alternative noise sampling strategies beyond purely random sampling on unstructured grids.

DDMs can also interact with Neural radiance fieldss (NERFs), which are analogous to PINNs without the physics-informed loss function.

Table 2
The main generative deep learning methods, the criterion on which their loss function is based, and the original or overview papers where they are further described. The abbreviations for the loss functions are: MaxLL = maximum likelihood; ELBO = evidence lower bound..
Source: Table inspired by Table 20.1 from Murphy [336].

Method (abbreviation)	Loss	Citation
Energy-based models	MaxLL	[337,338]
Normalising flows	MaxLL	[339,340]
Autoregressive models (ARMs)	MaxLL	[341,342]
Variational autoencoders (VAEs)	ELBO	[343,344]
Denoising diffusion models (DDMs)	ELBO	[345–347]
Generative adversarial networks (GANs)	Adversarial	[348,349]

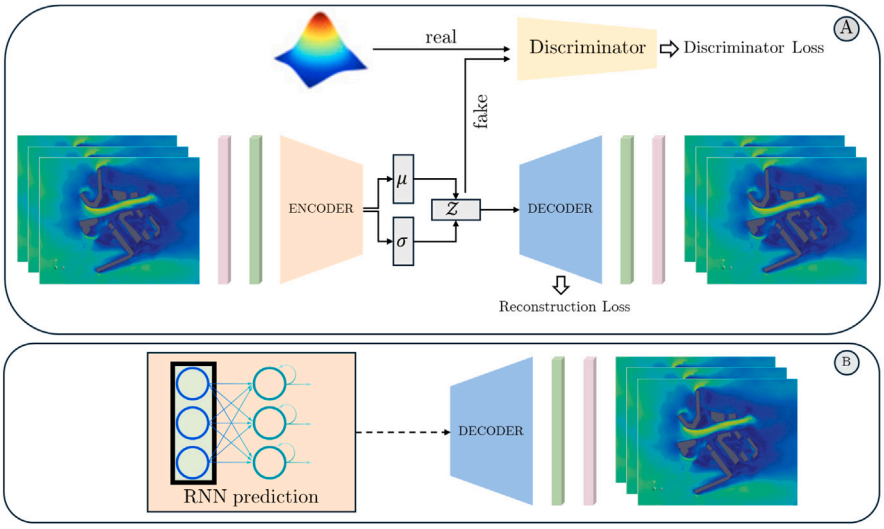


Fig. 20. Variational autoencoder and generative adversarial networks can be combined for forecasting on unstructured grids.
Source: Figure from Quilodrán-Casas and Arcucci [363].

In this context, DDMs can regularise the sampled fields [369] to more closely follow the DDM-generated data distribution. Furthermore, NERFs enable to train DDMs for 3D synthesis based on pre-trained 2D text-to-image models [370–373]. As NERFs represent functions of coordinates, they are inherently grid-independent and suitable for unstructured grids. However, the combined application of NERFs and DDMs can be computationally demanding, as both methods typically require substantial computational resources.

Neural operators aim to map between infinite-dimensional function spaces, allowing evaluation at arbitrary spatial positions [203,243]. To learn DDMs on such spaces, data at arbitrary positions can be interpolated to a structured grid, enabling the application of CNNs [374]. Additionally, convolutional layers can be replaced with implicit NERF-like representations [375], facilitating a grid-independent super-resolution operator. By integrating DDMs, neural operators can leverage the robust generative capabilities to efficiently handle complex and stochastic function mappings, potentially enhancing the accuracy and generalisation of mappings between function spaces.

Generative models have further been applied to three-dimensional meshes and point clouds, which are crucial for modelling entities such as the human body. VAEs and GANs are frequently utilised in this context [376–380]. Nonetheless, with their recent emergence in image generation, ARMs [381] and DDMs [382,383] are increasingly employed for the generation of such data.

In all these applications, generative models have emerged as powerful tools for handling unstructured data, offering a flexible approach to model complex and irregular datasets. These models excel in capturing the underlying distributions of unstructured data, enabling sophisticated applications such as advanced weather prediction, as demonstrated by models like GenCast from Google DeepMind. Furthermore, their adaptability to unstructured grids through techniques like graph neural networks and transformers enhances their utility in geospatial and scientific computing domains. However, the computational demands of these models, particularly when integrating with Neural Radiance Fields (NERFs) and neural operators, present significant challenges that require high computational resources and efficient algorithms. Despite these challenges, the ongoing research and development in optimising these models and expanding their applications suggest a bright future, promising enhanced accuracy, efficiency, and broader adoption across various scientific disciplines.

5. Public study cases and computational libraries

This section reviews the currently available open-access datasets and computational tools for physics problems involving unstructured grid data. The datasets can serve as benchmark test cases for assessing the performance of machine learning models in simulating dynamical systems on unstructured meshes. For the computational tools, we focus on generalisable libraries that include comprehensive user documentation and examples, making them adaptable to a wide range of applications.

5.1. Open-access datasets

For computational solid dynamics, challenges are faced in defining universal standard test cases due to the diverse nature of research areas. For example, in material mechanics, common tests for unstructured meshes include material tension/compression [384], bending [385], and fatigue tests [386,387], each tailored to specific material properties and behaviours. Shifting to CFD, we classify test cases with unstructured meshes into three distinct categories based on the temporal characteristics of the flow and the adaptability of the mesh. Each

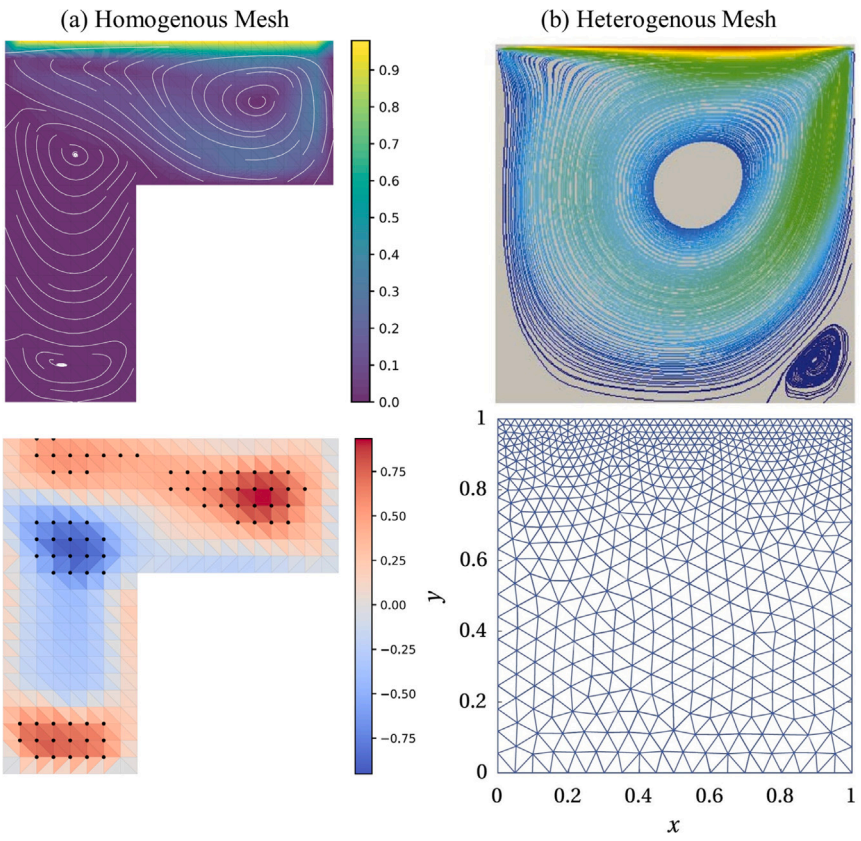


Fig. 21. Steady flow cases: Computational domain and computational mesh for lid-driven cavity flow of (a) homogeneous mesh [388] and (b) heterogeneous mesh [58].

category is designed to evaluate specific aspects of numerical methods and algorithms, reflecting the broad range of flow phenomena and challenges encountered in fluid dynamics simulations.

Steady flow with unstructured meshes. This category includes open-access datasets such as lid-driven cavity flow [58,388], isentropic vortex [201] and fluid dynamics around diverse shapes at low Reynolds number (Re) [226,389]. These datasets primarily utilise triangular and quadrilateral meshes, offering a range of mesh densities and configurations. For example, in the case of flow around a cylinder with Reynolds number (Re) ranging from 10 to 40, the mesh resolution is varied to accurately capture the boundary layer and wake regions. Finer meshes are employed near the cylinder surface and in areas with steep gradients to resolve complex flow features at different Re [390]. Additionally, these datasets provide comprehensive details on flow parameters, boundary conditions, and mesh specifications. Specifically, the lid-driven cavity flow datasets feature variations in Re, ranging from hundreds to thousands, enabling the study of flow behaviours from laminar to transitional regimes. Furthermore, the shape of the cavity can be modified to introduce additional complexity, as demonstrated in Fig. 21(a). These variations provide a versatile framework for comprehensively testing numerical accuracy, convergence, and the adaptability of computational methods to different geometrical configurations. The comparison between homogeneous and heterogeneous meshes is illustrated in Fig. 21, highlighting how uniform triangular meshes offer consistent resolution throughout the flow domain, while heterogeneous meshes adaptively refine regions of interest, such as areas with sharp gradients.

Time-dependent flow with fixed unstructured meshes. This category is pivotal for simulating dynamic flows using a static mesh, essential for assessing an algorithms' effectiveness in depicting time-varying fluid behaviours. Unlike the previous category, which focuses on steady-state simulations, this category deals with unsteady flows and incorporates time series data to capture transient phenomena. It is the most broadly utilised technique in CFD, dedicated to test cases that include the dynamics of flow around various shaped objects at high Re [221,391,392] such as cylinders, triangles [393] and airfoils [177, 394], alongside complex configurations [264,395] such as channel and backward-facing step flows, and extending to large-scale environmental phenomena [363,396]. The general mesh settings are similar to those in the 'steady flow with unstructured meshes' category, employing a mix of triangular and quadrilateral meshes [221], often in irregular formations. However, the main differences lie in the physical parameters, data structure and typically involve larger meshes with higher resolutions. Datasets in this category include more mesh elements to accurately capture complex, unsteady flow phenomena over time. This increased mesh density is essential for resolving finer details in high-fidelity simulations of dynamical flows. An illustrative example is the flow around a cylinder [389], as Re increases from 40 to 3900, the flow regime transitions from steady laminar to unsteady. This transition necessitates increasing the mesh complexity from 648 to 20,736 elements to accurately capture the complex flow dynamics. Fig. 22 exhibits typical examples of time-dependent flow with fixed meshes. The two-dimensional visualisation captures flow around a cylinder at a Re of 2300, using a mesh with 51,66 nodes across 1000 time steps. In a three-dimensional context, the figure portrays an urban air pollution simulation that incorporates a more extensive mesh with 100,040 nodes per dimension over 1000 time steps.

Time-dependent flow with adaptive meshes. This category features datasets where the mesh resolution is dynamically adjusted during simulations. This approach enhances simulation efficiency and accuracy by refining the mesh in regions with complex flow dynamics and coarsening it where less detail is needed, significantly reducing computational costs while capturing essential flow features. It is particularly effective for physical disturbances such as aerodynamic flows around objects [18,397], turbulence [398,399], and fluid–structure interactions [400,401], benefiting from the adaptive refinement's precision

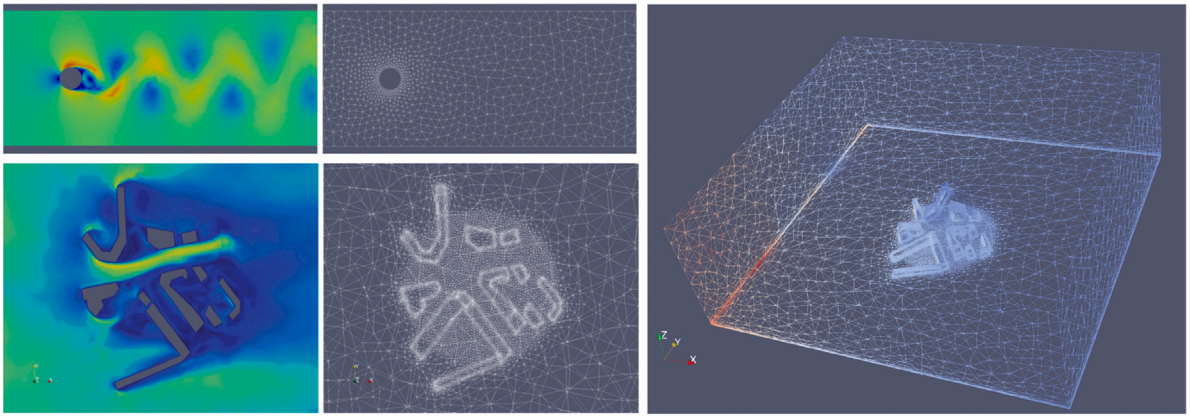


Fig. 22. Dynamic flow with fixed unstructured meshes: Computational domain and computational mesh for flow of water around a cylinder, and azimuthal and isometric view of urban air pollution simulation [363].

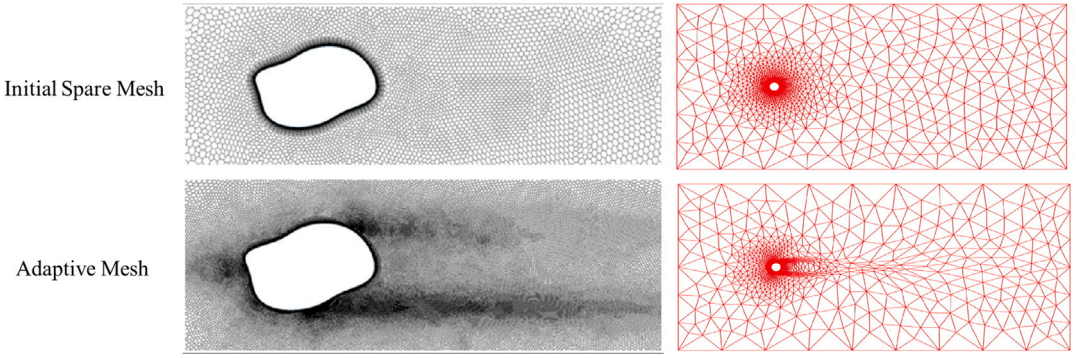


Fig. 23. Dynamic flow with adaptive unstructured meshes cases: Initial spare mesh and adaptive mesh for flow of water around a irregular object (left) [398] and cylinder (right) [402].

in complex flow areas. Specifically, adaptive meshing enables more intricate adjustments, which is particularly beneficial when dealing with scenarios that are similar but not identical. For instance, in aerodynamic studies, datasets often include simulations of flow around airfoils with slight variations in angle of attack or different Re [397]. These variations introduce changes in flow features such as boundary layer separation, vortex shedding, and pressure distribution, requiring the mesh to adapt dynamically to accurately capture these phenomena. The integration with multi-scale analysis is less common in this category, as it can be inherently related to multi-scale analysis by its nature. Fig. 23 shows the evolution from initial to adaptively refined meshes in two cases. Initially, heterogeneous meshes concentrate on refinement in regions anticipated to require higher resolution. Over time, through adaptive mesh refinement processes, these meshes undergo successive refinements, optimising the mesh density according to the evolving flow dynamics, such as areas of high gradient or vorticity.

By summarising the characteristics of the test cases in these three categories, we provide a reference for researchers applying ML to CFD. This overview may be used to assist in selecting appropriate benchmarks to validate model performance. Table 3 complements this discussion by presenting a comprehensive array of CFD test scenarios, detailing the specific characteristics of each test case along with available codes and datasets.

5.2. Open-access computational libraries

With benchmark cases established, the next key step is selecting suitable computational libraries for implementing and validating ML models in CFD. While building from scratch is feasible, leveraging mature libraries enhances development efficiency, reproducibility, and scalability by providing optimised solvers, automatic differentiation, and robust data handling. In this review, we focus exclusively on fully open-access computational libraries that are either specifically designed for, or have the potential to handle, computational problems involving unstructured grid data.

Computational libraries can be categorised into three main groups based on their methodological approach: PINN-based PDE solvers, neural operators, and geometric deep learning & graph-based ML. As summarised in Section 4.1 of this review paper, while PINNs embed physical laws directly into neural networks, neural operators learn solution operators for PDEs. These two methodologies often share overlapping computational frameworks and libraries. Meanwhile, geometric deep learning and graph-based methods provide an alternative approach by leveraging structured representations of unstructured spatial data to enhance predictive accuracy in complex fluid simulations.

5.2.1. PINN-based PDE solvers and neural operators

These solvers combine deep learning with physics-based constraints to solve PDEs, offering a data-efficient way to model complex physical systems. These libraries include DeepXDE, PhysicsNeMo (formerly known as Modulus and SimNet), NeuralPDE, and SciANN.

DeepXDE [410] is a flexible, multi-backend library (TensorFlow, PyTorch, JAX, PaddlePaddle) for PINNs, ResNets, and neural operators (e.g., FNO, DeepONet) that automates solver complexities, requiring only the problem's mathematical definition. It supports a wide range of domain geometries [411] (including primitive shapes, constructive solid geometry, and point clouds), boundary conditions (Dirichlet, Neumann, Robin, periodic, general), and automatic differentiation modes (reverse, forward, zero coordinate shift). Therefore, it is suitable for education and research, and new users only need to define the problem,

Table 3
Summary of test cases in fluid dynamics with unstructured meshes.

Category	Test case	Driven equation	Method	Solver	Adaptive mesh	Multi-scale	Available code/Dataset
Basic fluid flow	Advection	Advection equation	FDM	–	No	Yes	Data [403]
	Burgers' equation	Burgers' equation	FDM	–	No	Yes	[404]
	Wave problem	Wave equation/ Shallow water equation	FDM/FVM	OpenFOAM/⋯	No	Yes	Data [403]
Common physical phenomena	Lid-driven cavity	NS	FVM/FEM	FLUENT/OpenFOAM⋯	No	No	–
	Disturbance around objects	NS	RANS/FVM	SU2/FLUENT/⋯	Yes	No	Data [201,226] Data [18] Data [405,406]
	Channel flow	NS	RANS/FVM	OpenFOAM/Code Saturne/⋯	Yes	Yes	[264,407]
	Large-scale phenomena	–	RANS/FVM	PHIFLOW/FLUENT/⋯	Yes	Yes	[363,408] Data [409]

Note:
• The methods and solvers listed in this table represent a summary of findings from select publications.
• Each method and solver has its advantages and limitations, and should be chosen based on specific project requirements.
• FDM: Finite Difference Method; FEM: Finite Element Analysis; FVM: Finite Volume Analysis.

and then the solver can handle all the underlying complexities and solve the problem.

NeuralPDE [412] is a solver package for neural network-based solutions of xDEs (including ODEs, SDEs, RODEs, and PDEs), allowing for a broader range of problem formulations compared to classical approaches. As part of Julia's SciML ecosystem – a collection of tools for scientific computing with bindings to R, Python, and more – it focuses on multi-physics simulations and operator learning, supported by built-in error estimation. Its compatibility with Flux.jl and Lux.jl enables integration with GPU-powered machine learning layers, making it efficient for high-dimensional or complex PDE problems that leverage Julia's performance advantages.

PhysicsNeMo, developed by NVIDIA and formerly known as SimNet and Modulus, is a GPU-accelerated framework based on PyTorch designed for physics-informed modelling. PhysicsNeMo [413] contains a comprehensive library of state-of-art models designed for physics-ML applications, ideal for CFD, heat transfer and multi-physics optimisation tasks (e.g., airfoil design). It employs both PINNs and neural operators to efficiently tackle high-dimensional engineering problems. It provides a comprehensive end-to-end pipeline that covers the entire training process, starting from ingesting geometry, adding PDEs, and scaling the training to leverage the power of multinode GPUs. Additionally, it is designed to be highly extensible, allowing users to customise new functionality with minimal effort.

SciANN [414] is a flexible TensorFlow/Keras-based framework emphasising ease of use, supporting both PINNs [415] and neural operators [315]. It utilises TensorFlow and Keras capabilities, enabling seamless execution on both CPU and GPU for enhancing the computational efficiency. Applications range from surrogate modelling to general scientific and engineering simulations.

5.2.2. Geometric deep learning and graph-based ML

By leveraging graph-based representations, graph deep learning enables efficient learning on irregular data, capturing spatial and temporal dependencies as mentioned in Section 3.2 of this review paper. In this category, two public libraries are widely used: PyTorch-Geometric and Deep Graph Library.

PyTorch-Geometric (PyG) [416] is a general-purpose GNN library built on PyTorch, providing multi-GPU support and a flexible framework that allows easy implementation and extension to user-defined GNN architectures. PyG has been widely used in scientific machine learning, including fluid dynamics modelling [417], where its graph-based structure effectively handles unstructured data in computational simulations.

Deep Graph Library (DGL) [418] is an easy-to-use, high-performance, and scalable Python package for deep learning on graphs. Unlike framework-specific alternatives, DGL is framework-agnostic, allowing seamless integration with major deep learning frameworks such as PyTorch, Apache MXNet, and TensorFlow. It is designed for efficient training on large-scale graphs, supporting multi-GPU and distributed computing across multiple machines. In physical modelling, it has been applied to fluid dynamics simulations and PDE-driven processes, leveraging its flexible graph-based representation to efficiently handle unstructured spatial data in computational physics.

Despite the libraries mentioned above focusing on directly solving PDEs with neural networks, advanced meshing strategies remain equally crucial for accurately capturing complex solution behaviours. Pan et al. [321,322] proposed a fully automatic quadrilateral meshing framework powered by the Soft Actor-Critic RL algorithm, which learns an effective policy for element-by-element mesh generation. This approach bypasses the need for substantial hand-crafted rules, while still yielding high-quality meshes suitable for downstream PDE simulations—a complementary capability to the PINN and GNN libraries discussed earlier. Li et al. [310] proposed a novel framework, Geo-FNO, for solving PDEs on arbitrary geometries. Geo-FNO learns to deform potentially irregular input domains into a latent space with a uniform grid, where a FNO with fast Fourier transform is applied. This approach combines the computational efficiency of fast Fourier transform with the flexibility to handle complex geometries. Compared to standard numerical solvers, Geo-FNO achieves several times faster inference, and it also doubles the accuracy relative to existing machine learning-based PDE solvers, such as the standard FNO with direct interpolation.

A comprehensive summary of existing computational libraries is presented in Table 4. For each library, we detail the class of numerical methods it implements (such as PINNs, neural operators, or graph-based approaches), its primary software dependencies – including frameworks like TensorFlow, PyTorch, JAX, and Julia – and a range of application scenarios where the library has been employed. These include basic fluid flow (e.g., Burgers' or Poisson equations), flow simulations around complex geometries, and recent developments such as reinforcement learning for mesh generation.

6. Discussion and conclusion

This review highlights significant advancements and current achievements in the application of machine learning for handling unstructured grid data in computational physics. Notable strides include the adaptation of neural networks such as CNNs, GNNs and transformers for irregular geometries in complex dynamical systems. These models have demonstrated potential in various contexts and specific applications, including fluid dynamics, environmental modelling, and

Table 4
Overview of computational libraries for CFD and PDE solving appropriate for unstructured grid data.

Package	Method	Dependency	Application
DeepXDE	PINNs and Neural operators	TensorFlow, PyTorch, JAX, PaddlePaddle	Disturbance around objects [411]
NeuralPDE	XDE solver	Julia (Flux.jl, Lux.jl)	Burgers' equation, 2D Poisson equation [412]
PhysicsNeMo	PINNs & Neural operators	PyTorch	Disturbance around objects [419], Channel flow
SciANN	PINNs & Neural operators	TensorFlow, Keras	Burgers' equation, NavierStokes equations[a]
PyTorch-Geometric	Graph neural networks	PyTorch	Disturbance around objects [420,421]
Deep Graph Library	Graph-based ML	PyTorch, MXNet, TensorFlow	Disturbance around objects [222], Large-scale phenomena [422]
Geo-FNO	FNO	PyTorch	Disturbance around objects, Channel flow [310]
FreeMesh-RL	RL for mesh generation	PyTorch	Automatic quadrilateral mesh generation [321,322]

Note:
• XDEs = {ODEs, SDEs, RODEs, PDEs}.
• All libraries listed here support GPU computing.
[a] https://github.com/ehsanhaghighat/sciann-applications.

solid mechanics. In particular, they have shown substantial advantages over traditional physics-based solvers in terms of computational efficiency, parallel computing, and the integration of observational data. One may choose different neural network structures to work with, depending on the specific requirements and the availability of data or computational resources, as summarised at the end of Section 3.

On the other hand, different learning paradigms can be seamlessly applied to unstructured grid data, as discussed in this review. Physics-Informed Neural Networks provide a meshless solution by leveraging the power of auto-differentiation in neural networks and incorporating physics knowledge. Reinforcement learning, from another perspective, can be utilised to optimise mesh generation in computational systems. The challenge of data irregularity has also been addressed within popular generative AI paradigms, such as VAEs and diffusion models. These learning workflows are model-agnostic, meaning they can potentially be implemented using any of the neural network structures mentioned above.

However, significant challenges remain that limit the widespread adoption and scalability of these methods. A critical concern is computational efficiency when dealing with high-dimensional systems containing irregular data, as the resource demands of ML algorithms often scale exponentially with complexity, particularly for graph-based or transformer-type neural networks. Another pressing issue is the limited generalisability of developed ML models to unseen scenarios or data grids. While this challenge is common for structured data, it becomes even more pronounced for unstructured data points. Models trained on specific geometries or boundary conditions may struggle to adapt to novel configurations, topologies, or external forcing mechanisms, such as multi-physics interactions or dynamic environmental changes. When dealing with irregular data and varying geometries, such as simulation data with adaptive meshes, ensuring the model's generalisability and transferability to unseen scenarios is crucial—and remains a key challenge even for state-of-the-art learning approaches such as GNNs, PINNs, and Transformers. Furthermore, as a general challenge for machine learning algorithms, particularly in the context of complex dynamical systems, enhancing model interpretability to understand how predictions are made, and whether the model truly captures the underlying physics, is essential. In addition, deploying lightweight models to enable more efficient online inference is also a critical consideration across the various machine learning frameworks discussed in this review paper.

Furthermore, the lack of comprehensive and general-purpose datasets for benchmarking different approaches remains a significant challenge for the community. Existing datasets often target specific domains, making it difficult to objectively compare the performance of various ML techniques. The development of standardised, diverse datasets that encompass a wide range of conditions and topologies is essential for fostering progress. Similarly, the absence of suitable evaluation metrics for comparing systems that predict irregular data adds to the complexity. Metrics that account for irregularities in spatial resolution, topological structure, and multi-scale characteristics are needed to ensure fair and meaningful comparisons.

By tackling these challenges, interdisciplinary research combining computational physics, advanced machine learning, and applied mathematics can pave the way for more robust, scalable, and generalisable models, unlocking the full potential of ML in modelling unstructured grid data.

Notations

$\mathbf{x}^t$	Current state vector/field in the full space (t is the time)
$x^t_{i,j}$	One element (of index $\{i,j\}$) of the state field
$\tilde{\mathbf{x}}^t$	State vector in the reduced space
$\hat{\mathbf{x}}$	Predicted field vector
$\{\mathbf{X}\}$	An ensemble of state vectors
$\mathbf{i}$	Vector of node coordinates
$\tilde{n}$	Dimension of the compressed state
$\mathcal{L}$	Loss function in training neural networks
$\mathcal{D}$ and $\mathcal{E}$	Decoder and encoder for state variables ($\tilde{\mathbf{x}} = \mathcal{E}(\mathbf{x}), \mathbf{x} = \mathcal{D}(\tilde{\mathbf{x}})$)
$\mathbf{W}^l$	Weight matrix of the lth layer in the neural network
∇f	Gradient of a function f
$\frac{\partial f}{\partial x}$	Partial derivative of function f with respect to variable x
$\mathbb{E}[\mathbf{x}]$	Expected value of a random vector $\mathbf{x}$
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
$\mathbf{U}, \mathbf{V}$	Unitary matrices in POD
$\mathbf{\Sigma}$	Diagonal matrix in POD
$\mathbf{I}$	Identity matrix
ϕ_s	POD vectors
$\|\mathbf{x}\|$	Norm of vector $\mathbf{x}$
$\mathbf{A}^G$	Adjacency matrix of graph $\mathbf{G}$
$\mathbf{V}^G$	Node-level features of graph $\mathbf{G}$
$\mathbf{E}^G$	Edge-level features of graph $\mathbf{G}$
Q, K, V	Attention head in transformer
N_b	Total number of collocation points on the boundary
N_p	Total number of collocation points inside the domain
N_d	Total number of data points where direct measurements are available
N_v	Total number of data points in the validation dataset
$\mathcal{L}_{PDE}$	PDE loss
$\mathcal{L}_{BC}$	Boundary condition loss
$\mathcal{L}_{IC}$	Initial condition loss
$\mathcal{B}$	Geometry boundary

$\mathcal{R}$	Reward for assessing the quality of a new mesh element
Q^v	Action value in reinforcement learning
G^r	Cumulative rewards in reinforcement learning

Acronyms

2D Two-dimensional

AE Autoencoder

ARM Autoregressive model

CFD Computational fluid dynamics

CNN Convolutional neural network

DDM Denoising diffusion model

DMD Dynamic mode decomposition

ELBO Evidence lower bound

FNO Fourier neural operator

GAN Generative adversarial network

GNN Graph neural network

KNN K-Nearest neighbours

LSTM Long short-term memory

MAE Mean absolute error

ML Machine learning

MLP Multi layer perceptron

MSE Mean squared error

NERF Neural radiance fields

NLP Natural language processing

ODE Ordinary differential equation

PCA Principal component analysis

PDE Partial differential equation

PINN Physics-informed neural network

POD Proper orthogonal decomposition

EOF Empirical orthogonal functions

Re Reynolds number

RANS Reynolds-averaged Navier–Stokes

RF Random forest

RL Reinforcement learning

RNN Recurrent neural network

ROM Reduced-order modelling

SDE Stochastic differential equation

SINDy Sparse identification of nonlinear dynamics

SVD Singular value decomposition

VAE Variational Autoencoder

XGBoost EXtreme gradient boosting

CRediT authorship contribution statement

Sibo Cheng: Writing – original draft, Project administration, Methodology, Investigation, Conceptualization. **Marc Bocquet:** Writing – original draft, Methodology, Investigation. **Weiping Ding:** Writing – original draft, Methodology, Investigation. **Tobias Sebastian Finn:** Writing – original draft, Methodology, Investigation. **Rui Fu:** Writing – original draft, Methodology, Investigation. **Jinlong Fu:** Writing – original draft, Methodology, Investigation. **Yike Guo:** Writing – original draft, Methodology, Investigation. **Eleda Johnson:** Writing – original draft, Methodology, Investigation. **Siyi Li:** Writing – original draft, Methodology, Investigation. **Che Liu:** Writing – original draft, Methodology, Investigation. **Eric Newton Moro:** Writing – original draft, Methodology, Investigation. **Jie Pan:** Writing – original draft, Methodology, Investigation. **Matthew Piggott:** Writing – original draft, Methodology, Investigation. **Cesar Quilodran:** Writing – original draft, Methodology, Investigation. **Prakhar Sharma:** Writing – original draft, Methodology, Investigation. **Kun Wang:** Writing – original draft, Methodology, Investigation. **Dunhui Xiao:** Writing – original draft, Methodology, Investigation. **Xiao Xue:** Writing – original draft, Methodology, Investigation. **Yong Zeng:** Writing – original draft, Methodology, Investigation. **Mingrui Zhang:** Writing – original draft, Methodology, Investigation. **Hao Zhou:** Writing – original draft, Methodology, Investigation. **Kewei Zhu:** Writing – original draft, Methodology, Investigation. **Rossella Arcucci:** Writing – original draft, Methodology, Investigation.

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.

Acknowledgements

Sibo Cheng acknowledges the support of the French Agence Nationale de la Recherche (ANR) under reference ANR-22-CPJ2-0143-01. Weiping Ding acknowledges the support of the National Key R&D Plan of China under Grant 2024YFE0202700, the National Natural Science Foundation of China under Grant U2433216, and the Natural Science Foundation of Jiangsu Province, China under Grant BK20231337. Jinlong Fu acknowledges the financial support from Alexander von Humboldt Foundation. Dunhui Xiao acknowledges the Top Discipline Plan of Shanghai Universities-Class I and Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0100), National Key R&D Program of China (No. 2022YFE0208000). This work is supported in part by grants from the Shanghai Engineering Research Center for Blockchain Applications And Services (No. 19DZ2255100) and the Shanghai Institute of Intelligent Science and Technology, Tongji University. CEREA is a laboratory of Institut Pierre-Simon Laplace.

Data availability

No data was used for the research described in the article.

References

[1] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences, Rev. Modern Phys. 91 (4) (2019) 045002.

[2] J. Willard, X. Jia, S. Xu, M. Steinbach, V. Kumar, Integrating physics-based modeling with machine learning: A survey, 2020, pp. 1–34, arXiv preprint arXiv:2003.04919 1.

[3] K. Fukami, K. Fukagata, K. Taira, Super-resolution analysis via machine learning: a survey for fluid flows, Theor. Comput. Fluid Dyn. 37 (4) (2023) 421–444.

[4] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (1) (2020) 477–508.

[5] A. Katz, V. Sankaran, Mesh quality effects on the accuracy of CFD solutions on unstructured meshes, J. Comput. Phys. 230 (20) (2011) 7670–7686.

[6] F. Alauzet, A. Loseille, A decade of progress on anisotropic mesh adaptation for computational fluid dynamics, Comput.- Aided Des. 72 (2016) 13–39.

[7] M. Spiegel, T. Redel, Y.J. Zhang, T. Struffert, J. Hornegger, R.G. Grossman, A. Doerfler, C. Karmonik, Tetrahedral vs. polyhedral mesh size evaluation on flow velocity and wall shear stress for cerebral hemodynamic simulation, Comput. Methods Biomech. Biomed. Eng. 14 (01) (2011) 9–22.

[8] J. Southern, G.J. Gorman, M.D. Piggott, P.E. Farrell, Parallel anisotropic mesh adaptivity with dynamic load balancing for cardiac electrophysiology, J. Comput. Sci. 3 (1) (2012) 8–16.

[9] Q. Wang, W. Fang, R. de Richter, C. Peng, T. Ming, Effect of moving vehicles on pollutant dispersion in street canyon by using dynamic mesh updating method, J. Wind Eng. Ind. Aerodyn. 187 (2019) 15–25.

[10] D. Dundovic, J.G. Wallwork, S.C. Kramer, F. Gillet-Chaulet, R. Hock, M.D. Piggott, Anisotropic metric-based mesh adaptation for ice flow modelling in firedrake, EGUsphere 2024 (2024) 1–30.

[11] E. Benson, A. Mohammed, J. Gardell, S. Masich, E. Czeizler, P. Orponen, B. Högberg, DNA rendering of polyhedral meshes at the nanoscale, Nature 523 (7561) (2015) 441–444.

[12] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, J. Sci. Comput. 92 (3) (2022) 88.

[13] R. Loehner, D. Sharov, H. Luo, R. Ramamurti, Overlapping unstructured grids, in: 39th Aerospace Sciences Meeting and Exhibit, 2001, p. 439.

[14] C.E. Heaney, Y. Li, O.K. Matar, C.C. Pain, Applying convolutional neural networks to data on unstructured meshes with space-filling curves, Neural Netw. 175 (2024) 106198.

[15] K. Fukami, R. Maulik, N. Ramachandra, K. Fukagata, K. Taira, Global field reconstruction from sparse sensors with voronoi tessellation-assisted deep learning, Nat. Mach. Intell. 3 (11) (2021) 945–951.

[16] S. Cheng, C. Liu, Y. Guo, R. Arcucci, Efficient deep data assimilation with sparse observations and time-varying sensors, J. Comput. Phys. 496 (2024) 112581.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.

[18] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P.W. Battaglia, Learning mesh-based simulation with graph networks, in: International Conference on Learning Representations, 2021.

[19] R. Perera, V. Agrawal, Dynamic and adaptive mesh-based graph neural network framework for simulating displacement and crack fields in phase field models, Mech. Mater. 186 (2023) 104789.

[20] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, C. Shen, Twins: Revisiting the design of spatial attention in vision transformers, Adv. Neural Inf. Process. Syst. 34 (2021) 9355–9366.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[22] N. Geneva, N. Zabaras, Transformers for modeling physical systems, Neural Netw. 146 (2022).

[23] M. Raissi, P. Perdikaris, G. 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.

[24] 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.

[25] J. Donnelly, A. Daneshkhah, S. Abolfathi, Physics-informed neural networks as surrogate models of hydrodynamic simulators, Sci. Total Environ. 912 (2024) 168814.

[26] J. Donnelly, S. Abolfathi, J. Pearson, O. Chatrabgoun, A. Daneshkhah, Gaussian process emulation of spatio-temporal outputs of a 2D inland flood model, Water Res. 225 (2022) 119100.

[27] J. Donnelly, A. Daneshkhah, S. Abolfathi, Forecasting global climate drivers using Gaussian processes and convolutional autoencoders, Eng. Appl. Artif. Intell. 128 (2024) 107536.

[28] S.M. Ahmadi, S. Balahang, S. Abolfathi, Predicting the hydraulic response of critical transport infrastructures during extreme flood events, Eng. Appl. Artif. Intell. 133 (2024) 108573.

[29] C. Foucart, A. Charous, P.F. Lermusiaux, Deep reinforcement learning for adaptive mesh refinement, J. Comput. Phys. 491 (2023) 112381.

[30] C. Lorsung, A. Barati Farimani, Mesh deep q network: A deep reinforcement learning framework for improving meshes in computational fluid dynamics, AIP Adv. 13 (1) (2023).

[31] I. Kim, S. Kim, D. You, Non-iterative generation of an optimal mesh for a blade passage using deep reinforcement learning, Comput. Phys. Comm. 294 (2024) 108962.

[32] D.P. Kingma, M. Welling, et al., An introduction to variational autoencoders, Found. Trends® Mach. Learn. 12 (4) (2019) 307–392.

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.

[34] Y. Song, J. Sohl-Dickstein, D.P. Kingma, A. Kumar, S. Ermon, B. Poole, Score-based generative modeling through stochastic differential equations, 2020, arXiv preprint arXiv:2011.13456.

[35] F.-A. Croitoru, V. Hondru, R.T. Ionescu, M. Shah, Diffusion models in vision: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 45 (9) (2023) 10850–10869.

[36] C. Jacobsen, Y. Zhuang, K. Duraisamy, Cocogen: Physically-consistent and conditioned score-based generative models for forward and inverse problems, 2023, arXiv preprint arXiv:2312.10527.

[37] Y. Zhuang, S. Cheng, K. Duraisamy, Spatially-aware diffusion models with cross-attention for global field reconstruction with sparse observations, Comput. Methods Appl. Mech. Engrg. 435 (2025) 117623.

[38] I. Price, A. Sanchez-Gonzalez, F. Alet, T.R. Andersson, A. El-Kadi, D. Masters, T. Ewalds, J. Stott, S. Mohamed, P. Battaglia, et al., Probabilistic weather forecasting with machine learning, Nature (2024) 1–7.

[39] T.S. Finn, C. Durand, A. Farchi, M. Bocquet, P. Rampal, A. Carrassi, Generative diffusion for regional surrogate models from sea-ice simulations, J. Adv. Model. Earth Syst. 16 (10) (2024) e2024MS004395.

[40] A. Loseille, Unstructured mesh generation and adaptation, in: Handbook of Numerical Analysis, Vol. 18, Elsevier, 2017, pp. 263–302, http://dx.doi.org/10.1016/bs.hna.2016.10.004.

[41] N.J. Taylor, R. Haimes, Geometry modelling: Underlying concepts and requirements for computational simulation, in: 2018 Fluid Dynamics Conference, 2018, p. 3402.

[42] M.A. Park, W.L. Kleb, W.T. Jones, J.A. Krakos, T.R. Michal, A. Loseille, R. Haimes, J. Dannenhoffer, Geometry modeling for unstructured mesh adaptation, in: AIAA Aviation 2019 Forum, 2019, p. 2946.

[43] P.J. Frey, P.L. George, Mesh Generation: Application to Finite Elements, second ed., ISTE ; John Wiley & Sons, 2008.

[44] M.W. Bern, P.E. Plassmann, Mesh generation, Handb. Comput. Geom. 38 (2000).

[45] C.J. Budd, W. Huang, R.D. Russell, Adaptivity with moving grids, Acta Numer. 18 (2009) 111–241, http://dx.doi.org/10.1017/s0962492906400015.

[46] D. Mavriplis, Euler and Navier–Stokes Computations for Two-Dimensional Geometries Using Unstructured Meshes, Technical Report, 1990.

[47] M.D. Piggott, G.J. Gorman, C.C. Pain, P.A. Allison, A.S. Candy, B.T. Martin, M.R. Wells, A new computational framework for multi-scale ocean modelling based on adapting unstructured meshes, Internat. J. Numer. Methods Fluids 56 (8) (2008) 1003–1015, http://dx.doi.org/10.1002/fld.1663.

[48] Y. Ito, Challenges in unstructured mesh generation for practical and efficient computational fluid dynamics simulations, Comput. & Fluids 85 (2013) 47–52, http://dx.doi.org/10.1016/j.compfluid.2012.09.031.

[49] T. Marić, D.B. Kothe, D. Bothe, Unstructured un-split geometrical volume-of-fluid methods – A review, J. Comput. Phys. 420 (2020) 109695, http://dx.doi.org/10.1016/j.jcp.2020.109695.

[50] J.T. Hwang, J.R.R.A. Martins, An unstructured quadrilateral mesh generation algorithm for aircraft structures, Aerosp. Sci. Technol. 59 (2016) 172–182, http://dx.doi.org/10.1016/j.ast.2016.10.010.

[51] A. Slone, K. Pericleous, C. Bailey, M. Cross, Dynamic fluid–structure interaction using finite volume unstructured mesh procedures, Comput. Struct. 80 (5–6) (2002) 371–390, http://dx.doi.org/10.1016/S0045-7949(01)00177-8.

[52] T.J. Baker, Mesh generation: Art or science? Prog. Aerosp. Sci. 41 (1) (2005) 29–63.

[53] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev. 57 (4) (2015) 483–531.

[54] C.W. Rowley, S.T. Dawson, Model reduction for flow analysis and control, Annu. Rev. Fluid Mech. 49 (2017) 387–417.

[55] D. Xiao, F. Fang, A.G. Buchan, C.C. Pain, I.M. Navon, A. Muggeridge, Non-intrusive reduced order modelling of the Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 293 (2015) 522–541.

[56] C. Audouze, F. De Vuyst, P.B. Nair, Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations, Numer. Methods Partial Differential Equations 29 (5) (2013) 1587–1628.

[57] Y. Le Guennec, J.P. Brunet, F.Z. Daim, M. Chau, Y. Tourbier, A parametric and non-intrusive reduced order model of car crash simulation, Comput. Methods Appl. Mech. Engrg. 338 (2018) 186–207.

[58] J.S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, J. Comput. Phys. 363 (2018) 55–78.

[59] M. Guo, J.S. Hesthaven, Reduced order modeling for nonlinear structural analysis using Gaussian process regression, Comput. Methods Appl. Mech. Engrg. 341 (2018) 807–826.

[60] M. Guo, J.S. Hesthaven, Data-driven reduced order modeling for time-dependent problems, Comput. Methods Appl. Mech. Engrg. 345 (2019) 75–99.

[61] Z. Wang, D. Xiao, F. Fang, R. Govindan, C.C. Pain, Y. Guo, Model identification of reduced order fluid dynamics systems using deep learning, Internat. J. Numer. Methods Fluids 86 (4) (2018) 255–268.

[62] D. Xiao, C.E. Heaney, L. Mottet, F. Fang, W. Lin, I.M. Navon, Y. Guo, O.K. Matar, A.G. Robins, C.C. Pain, A reduced order model for turbulent flows in the urban environment using machine learning, Build. Environ. 148 (2019) 323–337.

[63] A.G. Buchan, C.C. Pain, F. Fang, I.M. Navon, A POD reduced-order model for eigenvalue problems with application to reactor physics, Internat. J. Numer. Methods Engrg. 95 (12) (2013) 1011–1032.

[64] F. Fang, C. Pain, I. Navon, G. Gorman, M. Piggott, P. Allison, P. Farrell, A. Goddard, A POD reduced order unstructured mesh ocean modelling method for moderate Reynolds number flows, Ocean. Model. 28 (1–3) (2009) 127–136.

[65] A. Manzoni, F. Salmoiraghi, L. Heltai, Reduced basis isogeometric methods (RB-IGA) for the real-time simulation of potential flows about parametrized NACA airfoils, Comput. Methods Appl. Mech. Engrg. 284 (2015) 1147–1180.

[66] Z. Luo, H. Li, P. Sun, J. An, I.M. Navon, A reduced-order finite volume element formulation based on POD method and numerical simulation for two-dimensional solute transport problems, Math. Comput. Simulation 89 (2013) 50–68.

[67] J. Du, F. Fang, C.C. Pain, I.M. Navon, J. Zhu, D.A. Ham, POD reduced-order unstructured mesh modeling applied to 2D and 3D fluid flow, Comput. Math. Appl. 65 (3) (2013) 362–379.

[68] F. Fang, T. Zhang, D. Pavlidis, C.C. Pain, A.G. Buchan, I.M. Navon, Reduced order modelling of an unstructured mesh air pollution model and application in 2D/3D urban street canyons, Atmos. Environ. 96 (2014) 96–106.

[69] R. Ştefănescu, I.M. Navon, POD/DEIM nonlinear model order reduction of an ADI implicit shallow water equations model, J. Comput. Phys. 237 (2013) 95–114.

[70] A.T. Mohan, D.V. Gaitonde, A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks, 2018, arXiv preprint arXiv:1804.09269.

[71] P. Wu, J. Sun, X. Chang, W. Zhang, R. Arcucci, Y. Guo, C.C. Pain, Data-driven reduced order model with temporal convolutional neural network, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112766.

[72] P.J. Schmid, Dynamic mode decomposition and its variants, Annu. Rev. Fluid Mech. 54 (2022) 225–254.

[73] J.N. Kutz, S.L. Brunton, B.W. Brunton, J.L. Proctor, Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems, SIAM, 2016.

[74] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28.

[75] J.H. Tu, Dynamic Mode Decomposition: Theory and Applications (Ph.D. thesis), Princeton University, 2013.

[76] K. Taira, S.L. Brunton, S.T. Dawson, C.W. Rowley, T. Colonius, B.J. McKeon, O.T. Schmidt, S. Gordeyev, V. Theofilis, L.S. Ukeiley, Modal analysis of fluid flows: An overview, Aiaa J. 55 (12) (2017) 4013–4041.

[77] B.O. Koopman, Hamiltonian systems and transformation in Hilbert space, Proc. Natl. Acad. Sci. 17 (5) (1931) 315–318.

[78] I. Mezić, Spectral properties of dynamical systems, model reduction and decompositions, Nonlinear Dynam. 41 (1–3) (2005) 309–325.

[79] I. Mezić, Analysis of fluid flows via spectral properties of the Koopman operator, Annu. Rev. Fluid Mech. 45 (1) (2013) 357–378.

[80] C.W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, D.S. Henningson, Spectral analysis of nonlinear flows, J. Fluid Mech. 641 (2009) 115–127.

[81] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[82] S. Cheng, C. Quilodrán-Casas, S. Ouala, A. Farchi, C. Liu, P. Tandeo, R. Fablet, D. Lucor, B. Iooss, J. Brajard, et al., Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review, IEEE/ CAA J. Autom. Sin. 10 (6) (2023) 1361–1387.

[83] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: A review, Neurocomputing 187 (2016) 27–48.

[84] C. Quilodrán-Casas, R. Arcucci, L. Mottet, Y. Guo, C. Pain, Adversarial autoencoders and adversarial LSTM for improved forecasts of urban air pollution simulations, 2021, arXiv preprint arXiv:2104.06297.

[85] T.R. Phillips, C.E. Heaney, P.N. Smith, C.C. Pain, An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion, Internat. J. Numer. Methods Engrg. 122 (15) (2021) 3780–3811.

[86] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, L. Carin, Variational autoencoder for deep learning of images, labels and captions, Adv. Neural Inf. Process. Syst. 29 (2016).

[87] S. Dutta, P. Rivera-Casillas, B. Styles, M.W. Farthing, Reduced order modeling using advection-aware autoencoders, Math. Comput. Appl. 27 (3) (2022) 34.

[88] F. Alsayyari, Z. Perkó, D. Lathouwers, J.L. Kloosterman, A nonintrusive reduced order modelling approach using proper orthogonal decomposition and locally adaptive sparse grids, J. Comput. Phys. 399 (2019) 108912.

[89] J. Baiges, R. Codina, I. Castanar, E. Castillo, A finite element reduced-order model based on adaptive mesh refinement and artificial neural networks, Internat. J. Numer. Methods Engrg. 121 (4) (2020) 588–601.

[90] K. Carlberg, Adaptive h-refinement for reduced-order models, Internat. J. Numer. Methods Engrg. 102 (5) (2015) 1192–1210.

[91] C. Pain, A. Umpleby, C. De Oliveira, A. Goddard, Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Comput. Methods Appl. Mech. Engrg. 190 (29–30) (2001) 3771–3796.

[92] Q. Wang, N. Ripamonti, J.S. Hesthaven, Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori–Zwanzig formalism, J. Comput. Phys. 410 (2020) 109402.

[93] R. Maulik, B. Lusch, P. Balaprakash, Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders, Phys. Fluids 33 (3) (2021).

[94] T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, K. Fukagata, Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow, Phys. Fluids 33 (2) (2021).

[95] P. Wu, S. Gong, K. Pan, F. Qiu, W. Feng, C. Pain, Reduced order model using convolutional auto-encoder with self-attention, Phys. Fluids 33 (7) (2021).

[96] R. Fu, D. Xiao, I.M. Navon, F. Fang, L. Yang, C. Wang, S. Cheng, A non-linear non-intrusive reduced order model of fluid flow by auto-encoder and self-attention deep learning methods, Internat. J. Numer. Methods Engrg. 124 (13) (2023) 3087–3111.

[97] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937.

[98] 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.

[99] U. Fasel, J.N. Kutz, B.W. Brunton, S.L. Brunton, Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control, Proc. R. Soc. A 478 (2260) (2022) 20210904.

[100] K. Fukami, T. Murata, K. Zhang, K. Fukagata, Sparse identification of nonlinear dynamics with low-dimensionalized flow representations, J. Fluid Mech. 926 (2021) A10.

[101] H. Gong, S. Cheng, Z. Chen, Q. Li, Data-enabled physics-informed machine learning for reduced-order modeling digital twin: application to nuclear reactor physics, Nucl. Sci. Eng. 196 (6) (2022) 668–693.

[102] J. Fu, D. Xiao, R. Fu, C. Li, C. Zhu, R. Arcucci, I.M. Navon, Physics-data combined machine learning for parametric reduced-order modelling of nonlinear dynamical systems in small-data regimes, Comput. Methods Appl. Mech. Engrg. 404 (2023) 115771.

[103] E. Fix, J.L. Hodges, Discriminatory analysis. Nonparametric discrimination: Consistency properties, Int. Stat. Rev./ Rev. Int. Stat. 57 (3) (1989) 238–247.

[104] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inform. Theory 13 (1) (1967) 21–27.

[105] K. Chomboon, P. Chujai, P. Teerarassamee, K. Kerdprasop, N. Kerdprasop, An empirical study of distance metrics for k-nearest neighbor algorithm, in: Proceedings of the 3rd International Conference on Industrial Application Engineering, Vol. 2, 2015, p. 4.

[106] C. Hoang, K. Chowdhary, K. Lee, J. Ray, Projection-based model reduction of dynamical systems using space–time subspace and machine learning, Comput. Methods Appl. Mech. Engrg. 389 (2022) 114341.

[107] Z. Gao, Y. Lin, X. Sun, X. Zeng, A reduced order method for nonlinear parameterized partial differential equations using dynamic mode decomposition coupled with k-nearest-neighbors regression, J. Comput. Phys. 452 (2022) 110907.

[108] E.M. Bollt, Model selection, confidence and scaling in predicting chaotic time-series, Int. J. Bifurc. Chaos 10 (06) (2000) 1407–1422.

[109] E.M. Bollt, Regularized forecasting of chaotic dynamical systems, Chaos Solitons Fractals 94 (2017) 8–15.

[110] D. Von Winterfeldt, W. Edwards, Decision analysis and behavioral research, 1986, pp. 63–69.

[111] S. Wille, A local solution adapted tri-tree multigrid generator and iterative equation solver for mixed finite element formulation of the Navier-Stokes equations, Comput. Methods Appl. Mech. Engrg. 131 (1–2) (1996) 109–132.

[112] B.A. Freno, K.T. Carlberg, Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations, Comput. Methods Appl. Mech. Engrg. 348 (2019) 250–296.

[113] O. Sagi, L. Rokach, Approximating xgboost with an interpretable decision tree, Inform. Sci. 572 (2021) 522–542.

[114] O. Sagi, L. Rokach, Explainable decision forest: Transforming a decision forest into an interpretable tree, Inf. Fusion 61 (2020) 124–138.

[115] Y. Izza, A. Ignatiev, J. Marques-Silva, On tackling explanation redundancy in decision trees, J. Artificial Intelligence Res. 75 (2022) 261–321.

[116] E. Farhi, S. Gutmann, Quantum computation and decision trees, Phys. Rev. A 58 (2) (1998) 915.

[117] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[118] T. Hackel, J.D. Wegner, K. Schindler, Contour detection in unstructured 3D point clouds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1610–1618.

[119] L. Auret, C. Aldrich, Change point detection in time series data with random forests, Control Eng. Pract. 18 (8) (2010) 990–1002.

[120] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[121] P. Kumar, M. Alruqi, H. Hanafi, P. Sharma, V.V. Wanatasanappan, Effect of particle size on second law of thermodynamics analysis of Al2O3 nanofluid: Application of XGBoost and gradient boosting regression for prognostic analysis, Int. J. Therm. Sci. 197 (2024) 108825.

[122] K. Zhu, S. Cheng, N. Kovalchuk, M. Simmons, Y.-K. Guo, O.K. Matar, R. Arcucci, Analyzing drop coalescence in microfluidic devices with a deep learning generative model, Phys. Chem. Chem. Phys. (2023).

[123] Y. Yan, X. Li, W. Sun, X. Fang, F. He, J. Tu, Semi-surrogate modelling of droplets evaporation process via XGBoost integrated CFD simulations, Sci. Total Environ. (2023) 164968.

[124] M. Seeger, Gaussian processes for machine learning, Int. J. Neural Syst. 14 (02) (2004) 69–106.

[125] F. Casenave, B. Staber, X. Roynard, MMGP: a mesh morphing Gaussian process-based machine learning method for regression of physical problems under nonparametrized geometrical variability, Adv. Neural Inf. Process. Syst. 36 (2024).

[126] J. Wang, A. Hertzmann, D.J. Fleet, Gaussian process dynamical models, Adv. Neural Inf. Process. Syst. 18 (2005).

[127] A. Damianou, M. Titsias, N. Lawrence, Variational Gaussian process dynamical systems, Adv. Neural Inf. Process. Syst. 24 (2011).

[128] C.K. Williams, Prediction with Gaussian processes: From linear regression to linear prediction and beyond, in: Learning in Graphical Models, Springer, 1998, pp. 599–621.

[129] R. Saraswat, D. Jhanwar, M. Gupta, Enhanced solar power forecasting using XG boost and PCA-based sky image analysis, Trait. Signal 41 (1) (2024).

[130] S. Cheng, Y. Jin, S.P. Harrison, C. Quilodrán-Casas, I.C. Prentice, Y.-K. Guo, R. Arcucci, Parameter flexible wildfire prediction using machine learning techniques: Forward and inverse modelling, Remote. Sens. 14 (13) (2022) 3228.

[131] H. Gong, S. Cheng, Z. Chen, Q. Li, C. Quilodrán-Casas, D. Xiao, R. Arcucci, An efficient digital twin based on machine learning SVD autoencoder and generalised latent assimilation for nuclear reactor physics, Ann. Nucl. Energy 179 (2022) 109431.

[132] J. Kocijan, R. Murray-Smith, C.E. Rasmussen, A. Girard, Gaussian process model based predictive control, in: Proceedings of the 2004 American Control Conference, Vol. 3, IEEE, 2004, pp. 2214–2219.

[133] J.M. Wang, D.J. Fleet, A. Hertzmann, Gaussian process dynamical models for human motion, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2) (2007) 283–298.

[134] L. Csató, M. Opper, Sparse on-line Gaussian processes, Neural Comput. 14 (3) (2002) 641–668.

[135] D. Nguyen-Tuong, J. Peters, M. Seeger, Local Gaussian process regression for real time online model learning, Adv. Neural Inf. Process. Syst. 21 (2008).

[136] L. Alzubaidi, J. Zhang, A.J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M.A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, J. Big Data 8 (2021) 1–74.

[137] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects, IEEE Trans. Neural Netw. Learn. Syst. 33 (12) (2021) 6999–7019.

[138] A. Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, Phys. D: Nonlinear Phenom. 404 (2020) 132306.

[139] A. Dhillon, G.K. Verma, Convolutional neural network: a review of models, methodologies and applications to object detection, Prog. Artif. Intell. 9 (2) (2020) 85–112.

[140] S. Sony, K. Dunphy, A. Sadhu, M. Capretz, A systematic review of convolutional neural network-based structural condition assessment techniques, Eng. Struct. 226 (2021) 111347.

[141] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. 64 (2019) 525–545.

[142] X. Jin, K. Liu, J. Jiang, T. Xu, Z. Ding, X. Hu, Y. Huang, D. Zhang, S. Li, K. Xue, et al., Pattern recognition of distributed optical fiber vibration sensors based on resnet 152, IEEE Sens. J. (2023).

[143] D. Coscia, L. Meneghetti, N. Demo, G. Stabile, G. Rozza, A continuous convolutional trainable filter for modelling unstructured data, Comput. Mech. 72 (2) (2023) 253–265.

[144] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[145] I.C. Duta, L. Liu, F. Zhu, L. Shao, Improved residual networks for image and video recognition, in: 2020 25th International Conference on Pattern Recognition, ICPR, IEEE, 2021, pp. 9415–9422.

[146] Z. Feng, C. Du, J. An, Z. Zhang, Handle heavy workload in penalty-based machine-type communication: Using ResNet, IEEE Wirel. Commun. Lett. (2024).

[147] M. Kaur, A. Mohta, A review of deep learning with recurrent neural network, in: 2019 International Conference on Smart Systems and Inventive Technology, ICSSIT, IEEE, 2019, pp. 460–465.

[148] M. Beiran, C.A. Spencer-Salmon, K. Rajan, A 'programming' framework for recurrent neural networks, Nat. Mach. Intell. 5 (6) (2023) 570–571.

[149] E.Z. Chen, P. Wang, X. Chen, T. Chen, S. Sun, Pyramid convolutional RNN for MRI image reconstruction, IEEE Trans. Med. Imaging 41 (8) (2022) 2033–2047.

[150] J. Farooq, M.A. Bazaz, D. Rafiq, Multiscale autoencoder-RNN architecture for mitigating error accumulation in long-term forecasting, in: 2023 International Conference on Emerging Techniques in Computational Intelligence, ICETCI, IEEE, 2023, pp. 271–275.

[151] R.K. Inapakurthi, S.S. Miriyala, K. Mitra, Deep learning based dynamic behavior modelling and prediction of particulate matter in air, Chem. Eng. J. 426 (2021) 131221.

[152] M. Blandin, H.K. Connor, D.S. Öztürk, A.M. Keesee, V. Pinto, M.S. Mahmud, C. Ngwira, S. Priyadarshi, Multi-variate LSTM prediction of alaska magnetometer chain utilizing a coupled model approach, Front. Astron. Space Sci. 9 (2022) 846291.

[153] E. Adeli, L. Sun, J. Wang, A.A. Taflanidis, An advanced spatio-temporal convolutional recurrent neural network for storm surge predictions, Neural Comput. Appl. 35 (26) (2023) 18971–18987.

[154] A. Hernández, J.M. Amigó, Attention mechanisms and their applications to complex systems, Entropy 23 (3) (2021) 283.

[155] R. Sevilla, S. Perotto, K. Morgan, Mesh Generation and Adaptation: Cutting-Edge Techniques, Vol. 30, Springer Nature, 2022.

[156] Y. Xing, Q. Song, G. Cheng, Benefit of interpolation in nearest neighbor algorithms, SIAM J. Math. Data Sci. 4 (2) (2022) 935–956.

[157] F. Aurenhammer, R. Klein, Voronoi diagrams, Handb. Comput. Geom. 5 (10) (2000) 201–290.

[158] P. Lancaster, K. Salkauskas, Surfaces generated by moving least squares methods, Math. Comp. 37 (155) (1981) 141–158.

[159] C. De Boor, A Practical Guide to Splines, Vol. 2, Springer-Verlag, 1978, pp. 4135–4195.

[160] K. Hormann, Barycentric interpolation, in: Approximation Theory XIV: San Antonio 2013, Springer, 2014, pp. 197–218.

[161] M.S. Floater, K. Hormann, Surface parameterization: a tutorial and survey, in: Advances in Multiresolution for Geometric Modelling, Springer, 2005, pp. 157–186.

[162] R. Franke, Scattered data interpolation: tests of some methods, Math. Comp. 38 (157) (1982) 181–200.

[163] M.D. Buhmann, Radial basis functions, Acta Numer. 9 (2000) 1–38.

[164] N. Cressie, The origins of kriging, Math. Geol. 22 (1990) 239–252.

[165] M. Oliver, R. Webster, A tutorial guide to geostatistics: Computing and modelling variograms and kriging, Catena 113 (2014) 56–69.

[166] H. Omre, K.B. Halvorsen, The Bayesian bridge between simple and universal kriging, Math. Geol. 21 (1989) 767–786.

[167] N. Cressie, Spatial prediction and ordinary kriging, Math. Geol. 20 (1988) 405–421.

[168] D. Zimmerman, C. Pavlik, A. Ruggles, M.P. Armstrong, An experimental comparison of ordinary and universal kriging and inverse distance weighting, Math. Geol. 31 (1999) 375–390.

[169] P.E. Farrell, M.D. Piggott, C.C. Pain, G.J. Gorman, C.R. Wilson, Conservative interpolation between unstructured meshes via supermesh construction, Comput. Methods Appl. Mech. Engrg. 198 (33–36) (2009) 2632–2642.

[170] D.C. Thomas, L. Engvall, S.K. Schmidt, K. Tew, M.A. Scott, U-splines: Splines over unstructured meshes, Comput. Methods Appl. Mech. Engrg. 401 (2022) 115515.

[171] D.M. Barker, W. Huang, Y.-R. Guo, A. Bourgeois, Q. Xiao, A three-dimensional variational data assimilation system for MM5: Implementation and initial results, Mon. Weather Rev. 132 (4) (2004) 897–914.

[172] H. Elbern, H. Schmidt, Ozone episode analysis by four-dimensional variational chemistry data assimilation, J. Geophys. Res.: Atmos. 106 (D4) (2001) 3569–3590.

[173] H. Liu, Y.-S. Ong, X. Shen, J. Cai, When Gaussian process meets big data: A review of scalable GPs, IEEE Trans. Neural Netw. Learn. Syst. 31 (11) (2020) 4405–4423.

[174] J. Li, A.D. Heap, A. Potter, J.J. Daniell, Application of machine learning methods to spatial interpolation of environmental variables, Environ. Model. Softw. 26 (12) (2011) 1647–1659.

[175] B. Liu, M. Wang, H. Foroosh, M. Tappen, M. Pensky, Sparse convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 806–814.

[176] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, R. Urtasun, Deep parametric continuous convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2589–2597.

[177] M. Xu, S. Song, X. Sun, W. Zhang, A convolutional strategy on unstructured mesh for the adjoint vector modeling, Phys. Fluids 33 (3) (2021).

[178] M.-Y. Wu, J.-Z. Peng, Z.-M. Qiu, Z.-H. Chen, Y.-B. Li, W.-T. Wu, Computationally effective estimation of supersonic flow field around airfoils using sparse convolutional neural network, Fluid Dyn. Res. 55 (3) (2023) 035504.

[179] K. Wen, L. Guo, Z. Xia, S. Cheng, J. Chen, A hybrid simulation method integrating CFD and deep learning for gas-liquid bubbly flow, Chem. Eng. J. 495 (2024) 153515.

[180] A. Palha, L. Manickathan, C.S. Ferreira, G. van Bussel, A hybrid Eulerian-Lagrangian flow solver, 2015, arXiv preprint arXiv:1505.03368.

[181] J. Hou, Y. Wang, B. Hou, J. Zhou, Q. Tian, Spatial simulation and prediction of air temperature based on CNN-LSTM, Appl. Artif. Intell. 37 (1) (2023) 2166235.

[182] X. Cao, K. Wu, X. Geng, Q. Guan, Field detection of indoor fire threat situation based on LSTM-kriging network, J. Build. Eng. 84 (2024) 108686.

[183] D.F. Watson, Computing the n-dimensional delaunay tessellation with application to Voronoi polytopes, Comput. J. 24 (2) (1981) 167–172.

[184] H. Wang, H. Zhou, S. Cheng, Dynamical system prediction from sparse observations using deep neural networks with Voronoi tessellation and physics constraint, Comput. Methods Appl. Mech. Engrg. 432 (2024) 117339.

[185] M. Mohammadpour, H. Roshan, M. Arashpour, H. Masoumi, Machine learning assisted kriging to capture spatial variability in petrophysical property modelling, Mar. Pet. Geol. 167 (2024) 106967.

[186] T. Plötz, S. Roth, Neural nearest neighbors networks, Adv. Neural Inf. Process. Syst. 31 (2018).

[187] O. Obiols-Sales, A. Vishnu, N.P. Malaya, A. Chandramowlishwaran, SURFNet: Super-resolution of turbulent flows with transfer learning using small datasets, in: 2021 30th International Conference on Parallel Architectures and Compilation Techniques, PACT, IEEE, 2021, pp. 331–344.

[188] X.-H. Zhou, J.E. McClure, C. Chen, H. Xiao, Neural network–based pore flow field prediction in porous media using super resolution, Phys. Rev. Fluids 7 (7) (2022) 074302.

[189] A. Kashefi, D. Rempe, L.J. Guibas, A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries, Phys. Fluids 33 (2) (2021).

[190] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 652–660.

[191] Y. Li, Z. Baorong, X. Xiaohong, L. Zijun, Application of a semivariogram based on a deep neural network to ordinary kriging interpolation of elevation data, PLOS One 17 (4) (2022) e0266942.

[192] H. Kang, Z. Tian, G. Chen, L. Li, T. Wang, Application of POD reduced-order algorithm on data-driven modeling of rod bundle, Nucl. Eng. Technol. 54 (1) (2022) 36–48.

[193] Y. Xu, Y. Sha, C. Wang, W. Cao, Y. Wei, Comparative studies of predictive models for unsteady flow fields based on deep learning and proper orthogonal decomposition, Ocean Eng. 272 (2023) 113935.

[194] H. Qian, P. Ma, S. Gao, Y. Song, Soft reordering one-dimensional convolutional neural network for credit scoring, Knowl.-Based Syst. 266 (2023) 110414.

[195] S. Cheng, J. Chen, C. Anastasiou, P. Angeli, O.K. Matar, Y.-K. Guo, C.C. Pain, R. Arcucci, Generalised latent assimilation in heterogeneous reduced spaces with machine learning surrogate models, J. Sci. Comput. 94 (1) (2023) 11.

[196] E. Cuthill, J. McKee, Reducing the bandwidth of sparse symmetric matrices, in: Proceedings of the 1969 24th National Conference, 1969, pp. 157–172.

[197] H. Wang, K. Gupta, L. Davis, A. Shrivastava, Neural space-filling curves, in: European Conference on Computer Vision, Springer, 2022, pp. 418–434.

[198] H. Gao, L. Sun, J.-X. Wang, PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. 428 (2021) 110079.

[199] X. Chen, J. Liu, Y. Pang, J. Chen, L. Chi, C. Gong, Developing a new mesh quality evaluation method based on convolutional neural network, Eng. Appl. Comput. Fluid Mech. 14 (1) (2020) 391–400.

[200] C. Lemeunier, F. Denis, G. Lavoué, F. Dupont, Representation learning of 3D meshes using an autoencoder in the spectral domain, Comput. Graph. 107 (2022) 131–143.

[201] C.-B. Zhou, Q. Wang, Y.-X. Ren, Machine learning optimization of compact finite volume methods on unstructured grids, J. Comput. Phys. 500 (2024) 112746.

[202] L. Lingsch, M.Y. Michelis, E. de Bezenac, S.M. Perera, R.K. Katzschmann, S. Mishra, Beyond regular grids: Fourier-based neural operators on arbitrary domains, 2023, arXiv preprint arXiv:2305.19663.

[203] 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.

[204] C.Q. Casas, R. Arcucci, Y. Guo, Urban air pollution forecasts generated from latent space representation, in: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020.

[205] S. Fetni, T.Q.D. Pham, T.V. Hoang, H.S. Tran, L. Duchêne, X.-V. Tran, A.M. Habraken, Capabilities of auto-encoders and principal component analysis of the reduction of microstructural images; application on the acceleration of phase-field simulations, Comput. Mater. Sci. 216 (2023) 111820.

[206] C.-H. Pham, S. Ladjal, A. Newson, PCA-AE: Principal component analysis autoencoder for organising the latent space of generative networks, J. Math. Imaging Vision 64 (5) (2022) 569–585.

[207] E. Plaut, From principal subspaces to principal components with linear autoencoders, 2018, arXiv preprint arXiv:1804.10253.

[208] H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biol. Cybern. 59 (4) (1988) 291–294.

[209] M.M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

[210] P. Veličković, Everything is connected: Graph neural networks, Curr. Opin. Struct. Biol. 79 (2023) 102538.

[211] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[212] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, Vol. 30, 2017.

[213] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 29, Curran Associates, Inc., 2016.

[214] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 6861–6871.

[215] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, Stat 1050 (2018) 4.

[216] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks? in: International Conference on Learning Representations, 2022, URL: https://openreview.net/forum?id=F72ximsx7C1.

[217] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, PMLR, 2017, pp. 1263–1272.

[218] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., Relational inductive biases, deep learning, and graph networks, 2018, arXiv preprint arXiv:1806.01261.

[219] F. Ogoke, K. Meidani, A. Hashemi, A.B. Farimani, Graph convolutional networks applied to unstructured flow field data, Mach. Learn.: Sci. Technol. 2 (4) (2021) 045020.

[220] F. De Avila Belbute-Peres, T. Economon, Z. Kolter, Combining differentiable PDE solvers and graph neural networks for fluid flow prediction, in: H. Daumé, A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 2402–2411.

[221] X. He, Y. Wang, J. Li, Flow completion network: Inferring the fluid dynamics from incomplete flow information using graph neural networks, Phys. Fluids 34 (8) (2022).

[222] Z. Li, K. Meidani, P. Yadav, A. Barati Farimani, Graph neural networks accelerated molecular dynamics, J. Chem. Phys. 156 (14) (2022).

[223] F. Pichi, B. Moya, J.S. Hesthaven, A graph convolutional autoencoder approach to model order reduction for parametrized PDEs, J. Comput. Phys. 501 (2024) 112762.

[224] O.M. Morrison, F. Pichi, J.S. Hesthaven, GFN: A graph feedforward network for resolution-invariant reduced operator learning in multifidelity applications, Comput. Methods Appl. Mech. Engrg. 432 (2024) 117458.

[225] W. Liu, M. Yagoubi, M. Schoenauer, Multi-resolution graph neural networks for PDE approximation, in: Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part III 30, Springer, 2021, pp. 151–163.

[226] J. Chen, E. Hachem, J. Viquerat, Graph neural networks for laminar flow prediction around random two-dimensional shapes, Phys. Fluids 33 (12) (2021).

[227] J. Suk, P. de Haan, P. Lippe, C. Brune, J.M. Wolterink, Mesh neural networks for SE (3)-equivariant hemodynamics estimation on the artery wall, Comput. Biol. Med. 173 (2024) 108328.

[228] S. Li, M. Zhang, M.D. Piggott, End-to-end wind turbine wake modelling with deep graph representation learning, Appl. Energy 339 (2023) 120928.

[229] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, in: H. Daumé, A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 8459–8468.

[230] W. Song, M. Zhang, J.G. Wallwork, J. Gao, Z. Tian, F. Sun, M.D. Piggott, J. Chen, Z. Shi, X. Chen, J. Wang, M2N: Mesh movement networks for PDE solvers, Adv. Neural Inf. Process. Syst. 35 (2022).

[231] S. Barwey, V. Shankar, V. Viswanathan, R. Maulik, Multiscale graph neural network autoencoders for interpretable scientific machine learning, J. Comput. Phys. 495 (2023) 112537.

[232] Z. Shi, X. Liang, J. Wang, LMC: Fast training of GNNs via subgraph sampling with provable convergence, 2023, arXiv preprint arXiv:2302.00924.

[233] J.-B. Cordonnier, A. Loukas, M. Jaggi, Multi-head attention: Collaborate instead of concatenate, 2020, arXiv preprint arXiv:2006.16362.

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

[235] B. Xu, Y. Zhou, X. Bian, Self-supervised learning based on transformer for flow reconstruction and prediction, Phys. Fluids 36 (2) (2024).

[236] L. Rampášek, M. Galkin, V.P. Dwivedi, A.T. Luu, G. Wolf, D. Beaini, Recipe for a general, powerful, scalable graph transformer, Adv. Neural Inf. Process. Syst. 35 (2022).

[237] M.J. Hutchinson, C.L. Lan, S. Zaidi, E. Dupont, Y.W. Teh, H. Kim, LieTransformer: Equivariant self-attention for lie groups, in: International Conference on Machine Learning, ICML, 2021.

[238] H. Qu, C. Li, sitian Qian, Particle transformer for jet tagging, in: International Conference on Machine Learning, ICML, 2022.

[239] S. Li, A. Robert, A.A. Faisal, M.D. Piggott, Learning to optimise wind farms with graph transformers, Appl. Energy 359 (2024) 122758.

[240] Z. Gao, X. Shi, H. Wang, Y. Zhu, Y.B. Wang, M. Li, D.-Y. Yeung, Earthformer: Exploring space-time transformers for earth system forecasting, Adv. Neural Inf. Process. Syst. 35 (2022).

[241] X. Han, H. Gao, T. Pfaff, J.-X. Wang, L.-P. Liu, Predicting physics in mesh-reduced space with temporal attention, Int. Conf. Learn. Represent. (2022).

[242] M. Zhang, C. Wang, S. Kramer, J.G. Wallwork, S. Li, J. Liu, X. Chen, M.D. Piggott, Towards universal mesh movement networks, Adv. Neural Inf. Process. Syst. 37 (2024).

[243] 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.

[244] S. Cao, Choose a transformer: Fourier or Galerkin, Adv. Neural Inf. Process. Syst. (NeurIPS) 34 (2021).

[245] G. Kissas, J. Seidman, L.F. Guilhoto, V.M. Preciado, G. J.Pappas, P. Perdikari, Learning operators with coupled attention, J. Mach. Learn. Res. (2022).

[246] T. Nguyen, M. Pham, T. Nguyen, K. Nguyen, S. Osher, N. Ho, Fourierformer: Transformer meets generalized fourier integral theorem, Adv. Neural Inf. Process. Syst. 35 (2022).

[247] Z. Hao, C. Ying, Z. Wang, H. Su, Y. Dong, S. Liu, Z. Cheng, J. Zhu, J. Song, GNOT: A general neural operator transformer for operator learning, in: International Conference on Machine Learning, ICML, 2023.

[248] Z. Li, D. Shu, A.B. Farimani, Scalable transformer for PDE surrogate modeling, Adv. Neural Inf. Process. Syst. (NeurIPS) 36 (2023).

[249] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901.

[250] K. He, X. Chen, S. Xie, Y. Li, P. Dollar, R. Girshick, Masked autoencoders are scalable vision learners, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022.

[251] 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.

[252] S. Subramanian, P. Harrington, K. Keutzer, W. Bhimji, D. Morozov, M. Mahoney, A. Gholami, Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior, Adv. Neural Inf. Process. Syst. (2023).

[253] G. Mialon, Q. Garrido, H. Lawrence, D. Rehman, Y. LeCun, B.T. Kiani, Self-supervised learning with Lie symmetries for partial differential equations, 2024, https://arxiv.org/abs/2307.05432.

[254] Z. Hao, C. Su, S. Liu, J. Berner, C. Ying, H. Su, A. Anandkumar, J. Song, J. Zhu, DPOT: auto-regressive denoising operator transformer for large-scale PDE pre-training, in: Proceedings of the 41st International Conference on Machine Learning, ICML '24, 2024.

[255] Y. Chen, J. Zhao, L. Huang, H. Chen, 3D mesh transformer: A hierarchical neural network with local shape tokens, Neurocomputing 514 (2022) 328–340.

[256] Y. Yoshiyasu, Deformable mesh transformer for 3D human mesh recovery, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 17006–17015.

[257] Y. Siddiqui, A. Alliegro, A. Artemov, T. Tommasi, D. Sirigatti, V. Rosov, A. Dai, M. Nießner, MeshGPT: Generating triangle meshes with decoder-only transformers, 2023, arXiv preprint arXiv:2311.15475.

[258] Y. Chen, J. Zhao, Q. Qiu, A transformer-based capsule network for 3D part–whole relationship learning, Entropy 24 (5) (2022) 678.

[259] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, H. Tan, Lrm: Large reconstruction model for single image to 3d, Int. Conf. Learn. Represent. (2024).

[260] L. Zhang, Z. Wang, Q. Zhang, Q. Qiu, A. Pang, H. Jiang, W. Yang, L. Xu, J. Yu, CLAY: A controllable large-scale generative model for creating high-quality 3D assets, ACM Trans. Graph. (SIGGRAPH) 2024 (2024).

[261] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: The Thirty-Second AAAI Conference on Artificial Intelligence, AAAI-18, 2018.

[262] L. Pandey, S. Wood, J. Wood, Are vision transformers more data hungry than newborn visual systems? Adv. Neural Inf. Process. Syst. 36 (2024).

[263] J. Decke, O. Wünsch, B. Sick, C. Gruhl, From structured to unstructured: a comparative analysis of computer vision and graph models in solving mesh-based PDEs, in: International Conference on Architecture of Computing Systems, Springer, 2024, pp. 82–96.

[264] L. Wang, Y. Fournier, J. Wald, Y. Mesri, A graph neural network-based framework to identify flow phenomena on unstructured meshes, Phys. Fluids 35 (7) (2023).

[265] C. Cai, J. Xiao, Y. Zou, X. He, Spatiotemporal reconstruction of unsteady bridge flow field via hierarchical graph neural networks with causal attention, Phys. Fluids 37 (1) (2025).

[266] S. Janny, A. Benetteau, N. Thome, M. Nadri, J. Digne, C. Wolf, EAGLE: Large-scale learning of turbulent fluid dynamics with mesh transformers, in: International Conference on Learning Representations, ICLR, 2023.

[267] R. Perera, V. Agrawal, Multiscale graph neural networks with adaptive mesh refinement for accelerating mesh-based simulations, Comput. Methods Appl. Mech. Engrg. 429 (2024) 117152.

[268] J. Jiang, G. Li, Y. Jiang, L. Zhang, X. Deng, TransCFD: A transformer-based decoder for flow field prediction, Eng. Appl. Artif. Intell. 123 (2023) 106340.

[269] M. Zhang, M.Z. Yousif, L. Yu, H.-C. Lim, A swin-transformer-based model for efficient compression of turbulent flow data, Phys. Fluids 35 (8) (2023).

[270] J. Brandstetter, M. Welling, S. Pawar, Message passing neural PDE solvers, in: International Conference on Learning Representations, ICLR, 2022, URL: https://openreview.net/forum?id=R1gu8b4tPr.

[271] 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. (NeurIPS) 34 (2021) 26548–26560, URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/0c5b5e77e6c9b30c0a1d0d657d7b3252-Paper.pdf.

[272] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning physics-constrained deep surrogate models for high-dimensional stochastic problems, SIAM Rev. 63 (3) (2021) 685–731, http://dx.doi.org/10.1137/20M1336633.

[273] A.F. Psaros, X. Meng, Z. Zou, L. Guo, G.E. Karniadakis, Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons, J. Comput. Phys. 477 (2023) 111902.

[274] H. Lee, I.S. Kang, Neural algorithm for solving differential equations, J. Comput. Phys. 91 (1) (1990) 110–131.

[275] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.

[276] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, D. Mortari, Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations, Neurocomputing 457 (2021) 334–356.

[277] 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.

[278] X. Sun, B. Croke, S. Roberts, A. Jakeman, Comparing methods of randomizing sobol sequences for improving uncertainty of metrics in variance-based global sensitivity estimation, Reliab. Eng. Syst. Saf. 210 (2021) 107499.

[279] H. Faure, C. Lemieux, Generalized Halton sequences in 2008: A comparative study, ACM Trans. Model. Comput. Simul. (TOMACS) 19 (4) (2009) 1–31.

[280] T.-T. Wong, W.-S. Luk, P.-A. Heng, Sampling with Hammersley and Halton points, J. Graph. Tools 2 (2) (1997) 9–24.

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

[282] Z. Fang, A high-efficient hybrid physics-informed neural networks based on convolutional neural network, IEEE Trans. Neural Netw. Learn. Syst. 33 (10) (2021) 5514–5526.

[283] X. Xue, S. Wang, H.-D. Yao, L. Davidson, P.V. Coveney, Physics informed data-driven near-wall modelling for lattice Boltzmann simulation of high Reynolds number turbulent flows, Commun. Phys. 7 (1) (2024) 338.

[284] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T.Y. Hou, M. Tegmark, Kan: Kolmogorov-Arnold networks, 2024, arXiv preprint arXiv:2404.19756.

[285] A. Kashefi, T. Mukerji, Physics-informed PointNet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries, J. Comput. Phys. 468 (2022) 111510.

[286] A. Kashefi, Kolmogorov–Arnold PointNet: Deep learning for prediction of fluid fields on irregular geometries, Comput. Methods Appl. Mech. Engrg. 439 (2025) 117888.

[287] C. Robert, Monte Carlo statistical methods, 1999.

[288] L. Martino, V. Elvira, F. Louzada, Effective sample size for importance sampling based on discrepancy measures, Signal Process. 131 (2017) 386–401.

[289] M.A. Nabian, R.J. Gladstone, H. Meidani, Efficient training of physics-informed neural networks via importance sampling, Comput.- Aided Civ. Infrastruct. Eng. 36 (8) (2021) 962–977.

[290] C.L. Zhao, Solving Allen–Cahn and Cahn–Hilliard equations using the adaptive physics informed neural networks, Commun. Comput. Phys. 29 (3) (2020).

[291] L.M.U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, 2021.

[292] S. Shi, D. Liu, Z. Zhao, Non-Fourier heat conduction based on self-adaptive weight physics-informed neural networks, in: 2021 40th Chinese Control Conference, CCC, IEEE, 2021, pp. 8451–8456.

[293] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.

[294] A.D. Jagtap, K. Kawaguchi, G. Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proc. R. Soc. A 476 (2239) (2020) 20200334.

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

[296] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.

[297] P. Sharma, L. Evans, M. Tindall, P. Nithiarasu, Stiff-PDEs and physics-informed neural networks, Arch. Comput. Methods Eng. 30 (5) (2023) 2929–2958.

[298] P. Sharma, L. Evans, M. Tindall, P. Nithiarasu, Hyperparameter selection for physics-informed neural networks (PINNs)–application to discontinuous heat conduction problems, Numer. Heat Transfer B (2023) 1–15.

[299] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020).

[300] K. Shukla, A.D. Jagtap, G.E. Karniadakis, Parallel physics-informed neural networks via domain decomposition, J. Comput. Phys. 447 (2021) 110683.

[301] Z. Hu, A.D. Jagtap, G.E. Karniadakis, K. Kawaguchi, When do extended physics-informed neural networks (XPINNs) improve generalization? SIAM J. Sci. Comput. 44 (5) (2022) A3158–A3182.

[302] B. Moseley, A. Markham, T. Nissen-Meyer, Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, Adv. Comput. Math. 49 (4) (2023) 62.

[303] J. Oldenburg, F. Borowski, A. Öner, K.-P. Schmitz, M. Stiehm, Geometry aware physics informed neural network surrogate for solving Navier-Stokes equation (GAPINN), Adv. Model. Simul. Eng. Sci. 9 (1) (2022) 8.

[304] M. Kast, J.S. Hesthaven, Positional embeddings for solving PDEs with evolutional deep neural networks, J. Comput. Phys. (2024) 112986.

[305] C. Zeng, T. Burghardt, A.M. Gambaruto, RBF-PINN: Non-Fourier positional embedding in physics-informed neural networks, 2024, arXiv preprint arXiv:2402.08367.

[306] X. Huang, T. Alkhalifah, Efficient physics-informed neural networks using hash encoding, J. Comput. Phys. 501 (2024) 112760.

[307] L.Z. Zhao, X. Ding, B.A. Prakash, PINNsFormer: A transformer-based framework for physics-informed neural networks, 2023, arXiv preprint arXiv:2307.11833.

[308] S. Lee, Mesh-independent operator learning for partial differential equations, in: ICML 2022 2nd AI for Science Workshop, 2022.

[309] N.R. Franco, A. Manzoni, P. Zunino, Mesh-informed neural networks for operator learning in finite element spaces, J. Sci. Comput. 97 (2) (2023) 35.

[310] Z. Li, D.Z. Huang, B. Liu, A. Anandkumar, Fourier neural operator with learned deformations for PDEs on general geometries, J. Mach. Learn. Res. 24 (388) (2023) 1–26.

[311] S. Liu, Z. Hao, C. Ying, H. Su, Z. Cheng, J. Zhu, Nuno: A general framework for learning parametric PDEs with non-uniform data, in: International Conference on Machine Learning, PMLR, 2023, pp. 21658–21671.

[312] M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, A. Anandkumar, Neural operators with localized integral and differential kernels, 2024, arXiv preprint arXiv:2402.16845.

[313] 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.

[314] A. Jnini, H. Goordoyal, S. Dave, A. Korobenko, F. Vella, K. Fraser, Physics-constrained DeepONet for surrogate CFD models: a curved backward-facing step case, in: ICLR 2024 Workshop on AI4DifferentialEquations in Science, 2024.

[315] 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.

[316] B. Chen, C. Wang, W. Li, H. Fu, A hybrid decoder-DeepONet operator regression framework for unaligned observation data, Phys. Fluids 36 (2) (2024).

[317] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.

[318] Y. Zeng, G. Cheng, Knowledge-based free mesh generation of quadrilateral elements in two-dimensional domains, Comput.- Aided Civ. Infrastruct. Eng. 8 (4) (1993) 259–270.

[319] S. Yao, B. Yan, B. Chen, Y. Zeng, An ANN-based element extraction method for automatic mesh generation, Expert Syst. Appl. 29 (1) (2005) 193–206.

[320] Y. Zeng, S. Yao, Understanding design activities through computer simulation, Adv. Eng. Inform. 23 (3) (2009) 294–308.

[321] J. Pan, J. Huang, Y. Wang, G. Cheng, Y. Zeng, A self-learning finite element extraction system based on reinforcement learning, AI EDAM 35 (2) (2021) 180–208.

[322] J. Pan, J. Huang, G. Cheng, Y. Zeng, Reinforcement learning for automatic quadrilateral mesh generation: A soft actor–critic approach, Neural Netw. 157 (2023) 288–304.

[323] T. Sorgente, S. Biasotti, G. Manzini, M. Spagnuolo, A survey of indicators for mesh quality assessment, in: Computer Graphics Forum, Vol. 42, Wiley Online Library, 2023, pp. 461–483.

[324] H. Tong, K. Qian, E. Halilaj, Y.J. Zhang, SRL-assisted AFM: generating planar unstructured quadrilateral meshes with supervised and reinforcement learning-assisted advancing front method, J. Comput. Sci. 72 (2023) 102109.

[325] N. Wang, L. Zhang, X. Deng, Unstructured surface mesh smoothing method based on deep reinforcement learning, Comput. Mech. 73 (2) (2024) 341–364.

[326] J. Yang, T. Dzanic, B. Petersen, J. Kudo, K. Mittal, V. Tomov, J.-S. Camier, T. Zhao, H. Zha, T. Kolev, et al., Reinforcement learning for adaptive mesh refinement, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2023, pp. 5997–6014.

[327] A. Gillette, B. Keith, S. Petrides, Learning robust marking policies for adaptive mesh refinement, SIAM J. Sci. Comput. 46 (1) (2024) A264–A289.

[328] J. Yang, K. Mittal, T. Dzanic, S. Petrides, B. Keith, B. Petersen, D. Faissol, R. Anderson, Multi-agent reinforcement learning for adaptive mesh refinement, 2022, arXiv preprint arXiv:2211.00801.

[329] N. Freymuth, P. Dahlinger, T. Würth, S. Reisch, L. Kärger, G. Neumann, Swarm reinforcement learning for adaptive mesh refinement, Adv. Neural Inf. Process. Syst. 36 (2024).

[330] T. Dzanic, K. Mittal, D. Kim, J. Yang, S. Petrides, B. Keith, R. Anderson, DynAMO: Multi-agent reinforcement learning for dynamic anticipatory mesh optimization with applications to hyperbolic conservation laws, J. Comput. Phys. 506 (2024) 112924.

[331] H. Keramati, F. Hamdullahpur, M. Barzegari, Deep reinforcement learning for heat exchanger shape optimization, Int. J. Heat Mass Transfer 194 (2022) 123112.

[332] I. Kim, J. Chae, D. You, Automatic mesh generation for optimal CFD of a blade passage using deep reinforcement learning, 2024, Available at SSRN 4852465.

[333] B.C. DiPrete, R. Garimella, C.G. Cardona, N. Ray, Reinforcement learning for block decomposition of planar CAD models, Eng. Comput. (2024) 1–11.

[334] S. Mohamed, B. Lakshminarayanan, Learning in implicit generative models, 2016, arXiv preprint arXiv:1610.03483.

[335] J.M. Tomczak, Why deep generative modeling? in: Deep Generative Modeling, Springer, 2021, pp. 1–12.

[336] K.P. Murphy, Probabilistic Machine Learning: Advanced Topics, MIT Press, 2023.

[337] E.T. Jaynes, Information theory and statistical mechanics, Phys. Rev. 106 (4) (1957) 620.

[338] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, et al., A tutorial on energy-based learning, Predict. Struct. Data 1 (2006).

[339] V. Laparra, G. Camps-Valls, J. Malo, Iterative Gaussianization: from ICA to random rotations, IEEE Trans. Neural Netw. 22 (4) (2011) 537–549.

[340] G. Papamakarios, E. Nalisnick, D.J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference, J. Mach. Learn. Res. 22 (57) (2021) 1–64.

[341] B.J. Frey, Graphical Models for Machine Learning and Digital Communication, MIT Press, 1998.

[342] H. Larochelle, I. Murray, The neural autoregressive distribution estimator, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 29–37.

[343] D.P. Kingma, Auto-encoding variational Bayes, 2013, arXiv preprint arXiv:1312.6114.

[344] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: International Conference on Machine Learning, PMLR, 2014, pp. 1278–1286.

[345] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, S. Ganguli, Deep unsupervised learning using nonequilibrium thermodynamics, in: International Conference on Machine Learning, PMLR, 2015, pp. 2256–2265.

[346] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Adv. Neural Inf. Process. Syst. 33 (2020) 6840–6851.

[347] Y. Song, S. Ermon, Generative modeling by estimating gradients of the data distribution, in: Advances in Neural Information Processing Systems, 2019, pp. 11895–11907.

[348] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014).

[349] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 214–223.

[350] Z. Xiao, K. Kreis, A. Vahdat, Tackling the generative learning trilemma with denoising diffusion GANs, in: International Conference on Learning Representations, ICLR, 2022.

[351] M. Arjovsky, L. Bottou, Towards principled methods for training generative adversarial networks, 2017, arXiv preprint arXiv:1701.04862.

[352] B.D. Anderson, Reverse-time diffusion equation models, Stochastic Process. Appl. 12 (3) (1982) 313–326.

[353] P. Vincent, A connection between score matching and denoising autoencoders, Neural Comput. 23 (7) (2011) 1661–1674.

[354] Y. Song, S. Ermon, Improved techniques for training score-based generative models, Adv. Neural Inf. Process. Syst. 33 (2020) 12438–12448.

[355] T.S. Finn, L. Disson, A. Farchi, M. Bocquet, C. Durand, Representation learning with unconditional denoising diffusion models for dynamical systems, Nonlinear Process. Geophys. 31 (3) (2024) 409–431.

[356] S. Rühling Cachay, B. Zhao, H. Joren, R. Yu, DYffusion: A dynamics-informed diffusion model for spatiotemporal forecasting, Adv. Neural Inf. Process. Syst. 36 (2023) 45259–45287.

[357] B. Holzschuh, S. Vegetti, N. Thuerey, Solving inverse physics problems with score matching, Adv. Neural Inf. Process. Syst. 36 (2023).

[358] J. Song, C. Meng, S. Ermon, Denoising diffusion implicit models, 2020, arXiv preprint arXiv:2010.02502.

[359] P. Dhariwal, A. Nichol, Diffusion models beat gans on image synthesis, Adv. Neural Inf. Process. Syst. 34 (2021) 8780–8794.

[360] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10684–10695.

[361] T. Kadeethum, D. O'Malley, Y. Choi, H.S. Viswanathan, N. Bouklas, H. Yoon, Continuous conditional generative adversarial networks for data-driven solutions of poroelasticity with heterogeneous material properties, Comput. Geosci. 167 (2022) 105212.

[362] M. Cheng, F. Fang, I. Navon, C. Pain, A real-time flow forecasting with deep convolutional generative adversarial network: Application to flooding event in Denmark, Phys. Fluids 33 (5) (2021).

[363] C. Quilodrán-Casas, R. Arcucci, A data-driven adversarial machine learning for 3D surrogates of unstructured computational fluid dynamic simulations, Phys. A 615 (2023) 128564.

[364] N. Gao, H. Xue, W. Shao, S. Zhao, K.K. Qin, A. Prabowo, M.S. Rahaman, F.D. Salim, Generative adversarial networks for spatio-temporal data: A survey, ACM Trans. Intell. Syst. Technol. (TIST) 13 (2) (2022) 1–25.

[365] G. Yang, S. Sommer, A denoising diffusion model for fluid field prediction, 2023, arXiv preprint arXiv:2301.11661.

[366] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, et al., Learning skillful medium-range global weather forecasting, Science 382 (6677) (2023) 1416–1421.

[367] W. Peebles, S. Xie, Scalable diffusion models with transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 4195–4205.

[368] A. Vahdat, K. Kreis, J. Kautz, Score-based generative modeling in latent space, Adv. Neural Inf. Process. Syst. 34 (2021) 11287–11302.

[369] J. Wynn, D. Turmukhambetov, Diffusionerf: Regularizing neural radiance fields with denoising diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4180–4189.

[370] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, T.-Y. Lin, Magic3d: High-resolution text-to-3d content creation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 300–309.

[371] B. Poole, A. Jain, J.T. Barron, B. Mildenhall, Dreamfusion: Text-to-3d using 2d diffusion, 2022, arXiv preprint arXiv:2209.14988.

[372] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, J. Zhu, Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation, Adv. Neural Inf. Process. Syst. 36 (2024).

[373] H. Wang, X. Du, J. Li, R.A. Yeh, G. Shakhnarovich, Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 12619–12629.

[374] S. Bond-Taylor, C.G. Willcocks, ∞-Diff: Infinite resolution diffusion with subsampled mollified states, in: Interrnational Conference on Learning Representations, 2024.

[375] S. Gao, X. Liu, B. Zeng, S. Xu, Y. Li, X. Luo, J. Liu, X. Zhen, B. Zhang, Implicit diffusion models for continuous super-resolution, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 10021–10030.

[376] J. Liu, F. Yu, T. Funkhouser, Interactive 3D modeling with a generative adversarial network, in: 2017 International Conference on 3D Vision, 3DV, IEEE, 2017, pp. 126–134.

[377] Q. Tan, L. Gao, Y.-K. Lai, S. Xia, Variational autoencoders for deforming 3d mesh models, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5841–5850.

[378] S. Foti, B. Koo, D. Stoyanov, M.J. Clarkson, 3D shape variational autoencoder latent disentanglement via mini-batch feature swapping for bodies and faces, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 18730–18739.

[379] S.P. Tata, S. Mishra, 3D GANs and latent space: A comprehensive survey, 2023, arXiv preprint arXiv:2304.03932.

[380] S. Molnár, L. Tamás, Variational autoencoders for 3D data processing, Artif. Intell. Rev. 57 (2) (2024) 42.

[381] S. Chen, X. Chen, A. Pang, X. Zeng, W. Cheng, Y. Fu, F. Yin, B. Wang, J. Yu, G. Yu, et al., Meshxl: Neural coordinate field for generative 3d foundation models, Adv. Neural Inf. Process. Syst. 37 (2024) 97141–97166.

[382] A. Alliegro, Y. Siddiqui, T. Tommasi, M. Nießner, Polydiff: Generating 3d polygonal meshes with diffusion models, 2023, arXiv preprint arXiv:2312.11417.

[383] Z. Liu, Y. Feng, M.J. Black, D. Nowrouzezahrai, L. Paull, W. Liu, Meshdiffusion: Score-based generative 3d mesh modeling, 2023, arXiv preprint arXiv:2303.08133.

[384] D.C. Pagan, C.R. Pash, A.R. Benson, M.P. Kasemer, Graph neural network modeling of grain-scale anisotropic elastic behavior using simulated and measured microscale data, Npj Comput. Mater. 8 (1) (2022) 259.

[385] M. Maurizi, C. Gao, F. Berto, Predicting stress, strain and deformation fields in materials and structures with graph neural networks, Sci. Rep. 12 (1) (2022) 21834.

[386] J.A. Bomidi, N. Weinzapfel, T. Slack, S. Mobasher Moghaddam, F. Sadeghi, A. Liebel, J. Weber, T. Kreis, Experimental and numerical investigation of torsion fatigue of bearing steel, J. Tribol. 135 (3) (2013) 031103.

[387] T. Hanlon, J. Reimann, M.A. Soare, A. Singhal, J. Grande, M. Edgar, K.S. Aggour, J. Vinciquerra, Artificial intelligence enabled material behavior prediction, 2019, arXiv preprint arXiv:1906.05270.

[388] Y. Hu, B. Lei, V.M. Castillo, Graph learning in physical-informed mesh-reduced space for real-world dynamic systems, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 4166–4174.

[389] K. Tlales, K.-E. Otmani, G. Ntoukas, G. Rubio, E. Ferrer, Machine learning mesh-adaptation for laminar and turbulent flows: applications to high-order discontinuous Galerkin solvers, Eng. Comput. (2024) 1–23.

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

[391] C. Lav, R.D. Sandberg, J. Philip, Improvement in unsteady wake prediction through machine learning based RANS model training, in: Int. Symp. Unsteady Aerodyn. Aeroacoustics Aeroelasticity Turbomachines, Vol. 15, 2018, pp. 24–27.

[392] S.B. Reddy, A.R. Magee, R.K. Jaiman, J. Liu, W. Xu, A. Choudhary, A. Hussain, Reduced order model for unsteady fluid flows via recurrent neural networks, in: International Conference on Offshore Mechanics and Arctic Engineering, Vol. 58776, American Society of Mechanical Engineers, 2019, V002T08A007.

[393] C.Y. Lee, S. Cant, A grid-induced and physics-informed machine learning CFD framework for turbulent flows, Flow Turbul. Combust. 112 (2) (2024) 407–442.

[394] M. Nemati, A. Jahangirian, A data-driven machine learning approach for turbulent flow field prediction based on direct computational fluid dynamics database, J. Appl. Fluid Mech. 17 (1) (2023) 60–74.

[395] C.-L. Chang, Development of physics-based transition models for unstructured-mesh CFD codes using deep learning models, in: AIAA AVIATION 2021 FORUM, 2021, p. 2828.

[396] X. Shao, Z. Liu, S. Zhang, Z. Zhao, C. Hu, PIGNN-CFD: A physics-informed graph neural network for rapid predicting urban wind field defined on unstructured mesh, Build. Environ. 232 (2023) 110056.

[397] V. Ojha, G. Chen, K. Fidkowski, Initial mesh generation for solution-adaptive methods using machine learning, in: AIAA SCITECH 2022 Forum, 2022, p. 1244.

[398] K. Huang, M. Krügener, A. Brown, F. Menhorn, H.-J. Bungartz, D. Hartmann, Machine learning-based optimal mesh generation in computational fluid dynamics, 2021, arXiv preprint arXiv:2102.12923.

[399] K. Hasegawa, K. Fukami, T. Murata, K. Fukagata, Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes, Theor. Comput. Fluid Dyn. 34 (2020) 367–383.

[400] J.-Z. Peng, Y.-Z. Wang, S. Chen, Z.-H. Chen, W.-T. Wu, N. Aubry, Grid adaptive reduced-order model of fluid flow based on graph convolutional neural network, Phys. Fluids 34 (8) (2022).

[401] M.J. Whisenant, K. Ekici, Galerkin-free technique for the reduced-order modeling of fluid-structure interaction via machine learning, in: AIAA Scitech 2020 Forum, 2020, p. 1637.

[402] W. Tingfan, L. Xuejun, A. Wei, Z. Huang, L. Hongqiang, A mesh optimization method using machine learning technique and variational mesh adaptation, Chin. J. Aeronaut. 35 (3) (2022) 27–41.

[403] A. Dulny, A. Hotho, A. Krause, Dynabench: A benchmark dataset for learning dynamical systems from low-resolution data, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2023, pp. 438–455.

[404] H. Kumar, N. Yadav, A deep learning algorithm for solving generalized Burgers–Fisher and Burger's equations, Int. J. Appl. Comput. Math. 9 (5) (2023) 76.

[405] F.D.A. Belbute-Peres, T. Economon, Z. Kolter, Combining differentiable PDE solvers and graph neural networks for fluid flow prediction, in: International Conference on Machine Learning, PMLR, 2020, pp. 2402–2411.

[406] M.S. Selig, UIUC airfoil coordinates database, 1996, https://m-selig.ae.illinois.edu/ads.html. University of Illinois at Urbana-Champaign. (Accessed 17 April 2025).

[407] F. Archambeau, N. Méchitoua, M. Sakiz, Code saturne: A finite volume code for the computation of turbulent incompressible flows-industrial applications, Int. J. Finite Vol. 1 (1) (2004) http–www.

[408] H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz-Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers, et al., The ERA5 global reanalysis, Q. J. R. Meteorol. Soc. 146 (730) (2020) 1999–2049.

[409] Global ocean physics reanalysis, 2023, https://doi.org/10.48670/moi-00021. (Accessed 18 April 2025).

[410] 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.
[411] Y. Zhu, S. Zhao, Y. Zhou, H. Liang, X. Bian, An unstructured adaptive mesh refinement for steady flows based on physics-informed neural networks, 2024, arXiv preprint arXiv:2411.19200.
[412] K. Zubov, Z. McCarthy, Y. Ma, F. Calisto, V. Pagliarino, S. Azeglio, L. Bottero, E. Luján, V. Sulzer, A. Bharambe, et al., NeuralPDE: Automating physics-informed neural networks (PINNs) with error approximations, 2021, arXiv preprint arXiv:2107.09443.
[413] O. Hennigh, S. Narasimhan, M.A. Nabian, A. Subramaniam, K. Tangsali, Z. Fang, M. Rietmann, W. Byeon, S. Choudhry, NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework, in: International Conference on Computational Science, Springer, 2021, pp. 447–461.
[414] E. Haghighat, R. Juanes, SciANN: A keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113552.
[415] U. Waheed, E. Haghighat, T. Alkhalifah, C. Song, Q. Hao, Eikonal solution using physics-informed neural networks, in: EAGE 2020 Annual Conference & Exhibition Online, Vol. 2020, European Association of Geoscientists & Engineers, 2020, pp. 1–5.
[416] M. Fey, J.E. Lenssen, Fast graph representation learning with PyTorch geometric, 2019, arXiv preprint arXiv:1903.02428.
[417] F. Bonnet, J.A. Mazari, T. Munzer, P. Yser, P. Gallinari, An extensible benchmarking graph-mesh dataset for studying steady-state incompressible Navier-Stokes equations, 2022, arXiv preprint arXiv:2206.14709.
[418] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, et al., Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2019, arXiv preprint arXiv:1909.01315.
[419] W.B. Tay, T.-C. Liao-Yang, H.-R. Luo, K.-P. Chen, B.C. Khoo, Framework for optimization of two-element airfoil using nvidia modulus, a physics informed neural network solver, in: AIAA SCITECH 2024 Forum, 2024, p. 1984.
[420] J. Helwig, X. Zhang, H. Yu, S. Ji, A geometry-aware message passing neural network for modeling aerodynamics over airfoils, 2024, arXiv preprint arXiv:2412.09399.
[421] V. Travnikov, I. Plokhikh, R. Mullyadzhanov, Advancing graph neural network architecture for fluid flow and heat transfer surrogate modeling: Variable boundary conditions and geometry, Phys. Fluids 36 (12) (2024).
[422] X. Zhao, Y. Zhong, P. Li, RTG-GNN: A novel rock topology-guided approach for permeability prediction using graph neural networks, Geoenergy Sci. Eng. 243 (2024) 213358.