Journal of Computational Physics 501 (2024) 112762

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A graph convolutional autoencoder approach to model order reduction for parametrized PDEs

Federico Pichi [a,*], Beatriz Moya [b], Jan S. Hesthaven [a]

[a] Chair of Computational Mathematics and Simulation Science, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
[b] Aragon Institute in Engineering Research (I3A), University of Zaragoza, 50018 Zaragoza, Spain

ARTICLE INFO

Keywords:
Nonlinear reduced order modeling
Parametric PDEs
Graph neural networks
Convolutional autoencoder
Geometrical parameters

ABSTRACT

The present work proposes a framework for nonlinear model order reduction based on a Graph Convolutional Autoencoder (GCA-ROM). In the reduced order modeling (ROM) context, one is interested in obtaining real-time and many-query evaluations of parametric Partial Differential Equations (PDEs). Linear techniques such as Proper Orthogonal Decomposition (POD) and Greedy algorithms have been analyzed thoroughly, but they are more suitable when dealing with linear and affine models showing a fast decay of the Kolmogorov n-width. On one hand, the autoencoder architecture represents a nonlinear generalization of the POD compression procedure, allowing one to encode the main information in a latent set of variables while extracting their main features. On the other hand, Graph Neural Networks (GNNs) constitute a natural framework for studying PDE solutions defined on unstructured meshes. Here, we develop a non-intrusive and data-driven nonlinear reduction approach, exploiting GNNs to encode the reduced manifold and enable fast evaluations of parametrized PDEs. We show the capabilities of the methodology for several models: linear/nonlinear and scalar/vector problems with fast/slow decay in the physically and geometrically parametrized setting. The main properties of our approach consist of (i) high generalizability in the low-data regime even for complex behaviors, (ii) physical compliance with general unstructured grids, and (iii) exploitation of pooling and un-pooling operations to learn from scattered data.

1. Introduction

The numerical approximation of parameterized partial differential equations (PDEs) using standard high-fidelity techniques (i.e. Finite Element, Finite Volume, Spectral Element Methods) is often unfeasible in many-query and real-time contexts. The exploitation of Reduced Order Models (ROMs) [1,2] can reduce the computational resources (CPU time and storage) required for their analysis and simulation. During the last decades, due to increasing interest and efforts, ROMs have become a well-established class of methodologies based on solid mathematical foundations.

Among them, the Reduced Basis method [3,4] enables fast and reliable evaluations of the solution for new parameter values. To build the reduced space, which allows for these efficient computations, one usually exploits Proper Orthogonal Decomposition (POD), an SVD-based method to extract the principal components, or a Greedy algorithm, iteratively augmenting the space with a basis corresponding to the worst approximation in the parametric space.

* Corresponding author.
E-mail address: federico.pichi@epfl.ch (F. Pichi).

https://doi.org/10.1016/j.jcp.2024.112762
Received 12 May 2023; Received in revised form 6 November 2023; Accepted 8 January 2024

Available online 12 January 2024

These techniques allow decoupling the computation in two stages, offline and online, achieving good and predictable accuracy at a small computational cost in the online phase. Nevertheless, these are usually linear approaches, which are known for losing efficiency when the models are not easily reducible, or they are characterized by non-affine and nonlinear terms.

Exploring fast and efficient non-intrusive model order reduction techniques from a deep learning perspective [5–7] allows users to overcome some of the limitations of traditional approaches. Exploiting nonlinear machine learning methodologies helps in reaching a low-dimensional representation of the latent subspace, and to capture the correlations among the features due to its optimal capacity in learning patterns.

The autoencoder architecture, which has recently been the subject of many studies, is particularly well-suited in the ROM context. Indeed, it generalizes linear compression procedures, such as the POD. In the machine learning context, an autoencoder is constituted by nonlinear encoding and decoding structures connected through a bottleneck. The bottleneck identifies the latent dimension and plays the role of the reduced space, in which we compress the information of the high order system with the encoder. Then we "project" the reduced representation back to its original dimension thanks to the decoder. This is an example of an unsupervised learning task, where, instead of having input-output pairs, one seeks to approximate the identity operator by consecutively compressing and decompressing the features.

Although past works take advantage of Fully Connected Neural Networks (FCNNs) [8], studies have progressively adopted more optimized structures to exploit spatial and temporal correlations to perform more efficient training procedures [9–11]. An architecture that has been thoroughly investigated is Convolutional Neural Networks (CNNs), which exploit the spatial information of the data provided, hence detecting and learning patterns. Originally meant for image classification, they have recently been applied extensively to dynamical modeling and parametrized PDEs [12–14,5,6], showing great performance in several fluid dynamics benchmarks [15]. However, CNNs usually work with structured datasets, resembling the shape of an image composed by pixels. Although this is a common structure in the field of computer vision, it is a rare condition during the investigation of physical problems. One could attempt to employ Cartesian meshes to simulate dynamic behaviors [16], but irregular meshes are often required to adapt to the domain under investigation. Indeed, PDEs are often posed on complex and possibly parametrized domains, which involve unstructured meshes. A possible approach consists in reshaping these grids into image-like objects [6]. Despite its good performance, there is a lack of consistency in the interpretation of locality while convoluting nearby pixels. Indeed, it is hard to find an efficient ordering and reshaping which is physically consistent with the problem. As an alternative, interpolation and level set approximations reshape the data as structured meshes. However, this approach requires dense meshes not to lose information in the reshaping process, hence increasing the computational cost. For instance, in [17], the authors propose to perform a preprocessing step to find a 2D representation of the data with k-PCA. Nevertheless, algorithms progressively have evolved toward their adaptation to the geometry of the domain. Works such as mesh-informed neural networks [18] and continuous convolutional filters [19,20] suggest to re-think the neural network structures to incorporate geometrical information, and keep an inductive bias which is physically consistent.

Geometric deep learning [21] arises as a unifying theory to treat data exploiting information on geometry [22]. For example, an active line of research is devoted to generalizing convolutions to graphs and manifolds [23,24]. Here, we seek to preserve the underlying geometric structure of the data, known as *geometric priors*. The imposition of these provides a deeper understanding and a more accurate treatment of the dataset. Moreover, the additional knowledge included in the algorithm can compensate when dealing with scarce data, without drastically compromising the generalization properties.

This work is dedicated to the study of deep learning in model order reduction for parametric PDEs, developing and analyzing a data-driven and non-intrusive framework to deal with models defined on unstructured grids, capturing their geometric features through a combination of Convolutional Autoencoders and Graph Neural Networks (GNNs).

We take advantage of the straightforward interpretation of a field defined on a general mesh as a simple, undirected and connected graph, with the field as a feature.

Inspired by [5,6], we propose a modular architecture, namely Graph Convolutional Autoencoder for Reduced Order Modelling (GCA-ROM), see Fig. 1, which subsequently exploits:

(1) a graph-based layer to express an unstructured dataset;
(2) an encoder module compressing the information through:
    (i) spatial convolutional layers based on MoNet [24] to identify patterns between geometrically close regions;
    (ii) skip-connection operation, to keep track of the original information and help the learning procedure;
    (iii) a pooling operation, to down-sample the data to obtain smaller networks;
(3) a bottleneck map, connected to the encoder's output by means of a dense layer, which learns the latent behavior in a vector;
(4) a decoder module, connected to the latent vector through a dense layer, recovering the original data by applying the same operations as in the encoder, but in reverse order.

Using this approach, we study problems covering a wide spectrum of characteristics. Mainly, we focus on both linear and nonlinear parametrized PDEs with scalar and vector unknown fields, and analyze different configurations of the proposed architecture, to evaluate their advantages and drawbacks for several benchmarks in model order reduction. The methodology has been tested on a complex phenomenon exhibiting a bifurcating behavior (linked to the non-uniqueness of the solution). In this case, we exploit the bottleneck and a clustering technique to classify bifurcating regimes, and thus detect the bifurcation points.

We acknowledge the use of the open source software FEniCS [25] and RBniCS [26] to generate the dataset, and PyTorch Geometric [27] for the implementation and training of the graph convolutional autoencoder. The codes to reproduce the results and create a new test dataset from high-fidelity data are publicly available in the GitHub repository: https://github.com/fpichi/gca-rom.

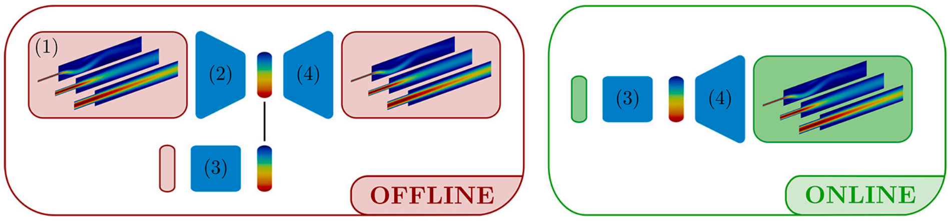


Fig. 1. Schematic representation of the GCA-ROM architecture. In the offline phase, a graph autoencoder is trained via unsupervised learning, while a multilayer perceptron learns the encoded representation through a mapping from the parametric space to the low dimensional space. In the online phase, the bottleneck map is evaluated, and the decoder is used to reconstruct the field. The numbers in the figure correspond to the indices referenced in its description.

This paper is structured as follows. After a brief overview of the GNNs literature, Section 2 is dedicated to the description of the model order reduction setting, differentiating intrusive projection-based methodologies and non-intrusive data-driven techniques. In Section 3, we detail the basic concepts of GNNs, and present our GCA-ROM architecture detailing all its components. Section 4 discusses the results obtained for different test cases. Finally, we state the conclusions and future research lines opening from the proposed methodology in Section 5.

1.1. Literature review on Graph Neural Networks

In recent years, Graph Neural Networks [28,29] have been an active area of research with rapid growth. GNNs are a specific kind of deep learning architectures capable of extracting information from a graph-structured dataset, including geometric configurations, relationships between nodes and connectivity, and features' behavior. They are used in different fields such as recommendation systems, social networks and to express molecular structures, for their ability to process non-euclidean data. Several tasks can be performed through GNNs

- at graph-level: where the goal is the property/label for the whole structure,
- at node-level: to infer the features at some specific location,
- at edge-level: to predict the existence of a given link/connection.

The information is usually propagated between nodes through two of the basic operations of GNNs: aggregation and message passing.

A key property of GNNs is that they preserve discrete symmetries, i.e. nodes do not have to be in any specific order when they are provided as input to the network. Consequently, the functions are permutation-invariant. This is a great advantage that allows generalizing common architectures such as Convolutional Neural Networks that require a fixed template/filter. Graph networks have been widely employed in recent research [30]. Notably, it has been especially successful in large deformation settings and complex simulations, drug discovery, and in general in scenarios where data is unstructured.

Focusing on our area of interest, graphs have been applied to the study of computational mechanics for the reconstruction of solution fields [31], the integration of complex systems in time [32], and combined with learning biases to preserve fundamental physical laws [33].

In the model order reduction context, we can discern between two classes. On the one hand, we find those that reduce the mesh to alleviate graph operations [34], and its edge-augmented version [35]. The work by Han et al. [36] fall into first class and focus on preserving the graph information by finding a coarser representation by collapsing nodes to speed up the simulation.

On the other hand, the second class of reduction methods is related to the classical ROM setting, with which a vector low-dimensional representation of the data is found. For example, in [37] the main goal is to deal with 3D representations for reconstruction and tracking. Graph U-nets [38] present an architecture of graph convolutions combined with down-sampling layers that lead to a low dimensional manifold. Ranjan et al. [39] introduce the so-called COMA network consisting of a combination of spectral Chebyshev convolutions [40] and down-sampling operations to generate new faces. In [41], the authors define the pooling operations based on varying kernels formulated at the mesh points. Also, the work of Kashefi et al. [42] presents the reconstruction of solution fields for a given geometry. In the context of parametrized PDEs, we acknowledge the recent work of Gruber et al. [43], which compared different ML approaches for data-driven ROMs, considering spectral convolutions without exploiting down-sampling procedures.

Indeed, the definition of down-sampling and up-sampling operations to coarsen and reconstruct domains is one of the most critical steps for an autoencoder-based architecture. These methods are widely studied for classification problems [44]. However, pooling, un-pooling, and inverse convolutions are not defined for non-euclidean domains. Although there are functions that perform an approximation of these operations [45], their application is still ongoing research. Currently, the most widely used reconstruction function for up-sampling is the one exploiting k-nearest neighbors [46]. Finally, we remark that since GNNs are still in their early stage of development, they offer much room for further investigations and developments.

2. Reduced order models for parametrized partial differential equations

In this section we review the standard setting of model order reduction in the context of parametrized partial differential equations. In particular, we will outline the main differences between classical approaches based on intrusive projections, as opposed to non-intrusive data-driven techniques.

2.1. *Intrusive projection-based methodologies*

In the field of computational mechanics, the numerical approximation of physical models described by PDEs is of utmost importance. Such models are usually characterized by the domain in which they are posed, geometric and/or physical parameters, and possibly nonlinear terms. The so-called high-fidelity techniques, such as the Finite Element method, involve a large number of degrees of freedom, preventing a fast resolution of the model under investigation. This issue is especially harmful in the parametrized context, when one is interested in recovering the solution for different physical/geometrical configurations, for many-query or real time investigations.

Reduced Order Models have been developed during the last decade to address this computational bottleneck. In particular, the Reduced Basis (RB) method [3,4] aims at constructing a low-dimensional space which captures the key features of the system. This way, one uses a Galerkin projection to project the model w.r.t. the basis spanning the low-dimensional manifold, to obtain a reduced system.

Given a suitable Hilbert space $\mathbb{X}_{\boldsymbol{\mu}} \doteq \mathbb{X}(\Omega(\boldsymbol{\mu}))$, we consider the nonlinear, steady, and parametrized problem: given $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^P$, seek $u(\boldsymbol{\mu}) \in \mathbb{X}_{\boldsymbol{\mu}}$ such that

$$F(u(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0 \qquad \text{in} \quad \mathbb{X}'_{\boldsymbol{\mu}}$$

where $\boldsymbol{\mu} \in \mathcal{P}$ is the parameter, $\Omega(\boldsymbol{\mu})$ the computational domain (possibly $\boldsymbol{\mu}$-dependent), and F expresses the PDE operator. A possible high-fidelity approach consists in a Galerkin projection of the associated weak-formulation over some finite dimensional subspace $\mathbb{X}^{\mathcal{N}}_{\boldsymbol{\mu}} \subset \mathbb{X}_{\boldsymbol{\mu}}$ of dimension $\mathcal{N}$.

Thanks to the Newton-Kantorovich method [47], the formulation of the linearized system at the k-th iteration reads: given $\boldsymbol{\mu} \in \mathcal{P}$, seek $\delta\mathbf{u}_{\mathcal{N}} \in \mathbb{R}^{\mathcal{N}}$ such that

$$\mathsf{J}_{\mathcal{N}}(\mathbf{u}^k_{\mathcal{N}}; \boldsymbol{\mu})\delta\mathbf{u}_{\mathcal{N}} = -\mathsf{F}_{\mathcal{N}}(\mathbf{u}^k_{\mathcal{N}}; \boldsymbol{\mu}) \qquad \text{in } \mathbb{R}^{\mathcal{N}} \tag{1}$$

where $\mathsf{F}_{\mathcal{N}} \in \mathbb{R}^{\mathcal{N}}$ and $\mathsf{J}_{\mathcal{N}} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ are the high-fidelity residual vector and the Jacobian matrix, respectively. Finally, we update the iteration as $\mathbf{u}^k_{\mathcal{N}} = \mathbf{u}^{k-1}_{\mathcal{N}} + \delta\mathbf{u}_{\mathcal{N}}$ until convergence.

Exploiting the same concept in the ROM context, one builds a basis $\{\boldsymbol{\zeta}_k\}_{k=1}^N \subset \mathbb{X}^{\mathcal{N}}_{\boldsymbol{\mu}}$ for the reduced space $\mathbb{X}^N_{\boldsymbol{\mu}}$, e.g. by means of the POD technique [48], to project the system once again, obtaining the following formulation: given $\boldsymbol{\mu} \in \mathcal{P}$, we seek $\delta\mathbf{u}_N \in \mathbb{R}^N$ such that

$$\mathsf{V}^T\mathsf{J}_{\mathcal{N}}(\mathsf{V}\mathbf{u}^k_N; \boldsymbol{\mu})\mathsf{V}\delta\mathbf{u}_N = -\mathsf{V}^T\mathsf{F}_{\mathcal{N}}(\mathsf{V}\mathbf{u}^k_N; \boldsymbol{\mu}) \qquad \text{in} \quad \mathbb{R}^N \tag{2}$$

where the basis matrix is encoded in $\mathsf{V} = [\boldsymbol{\zeta}_1|\dots|\boldsymbol{\zeta}_N] \in \mathbb{R}^{\mathcal{N}\times N}$, and we update the iteration as $\mathbf{u}^k_N = \mathbf{u}^{k-1}_N + \delta\mathbf{u}_N$. The reduced basis approximation is finally given by, $u_N(\boldsymbol{\mu}) = \sum_{k=1}^N u^{(k)}_N(\boldsymbol{\mu})\boldsymbol{\zeta}_k$, where $\mathbf{u}_N(\boldsymbol{\mu}) = \{u^{(k)}_N(\boldsymbol{\mu})\}_{k=1}^N$ is the reduced coefficient vector.

The POD basis can be extracted by exploiting the FE data, i.e. a set of solutions corresponding to different parameters $\{\boldsymbol{\mu}^i\}_{i=1}^{N_\mathrm{S}}$, the so-called snapshots $\{u_{\mathcal{N}}(\boldsymbol{\mu}^i)\}_{i=1}^{N_\mathrm{S}}$. The efficiency of the classical approach relies on the affine decomposition assumption, and on the independence of the online computations from the degrees of freedom $\mathcal{N}$. Unfortunately, these assumptions are not fulfilled in the general nonlinear context, since (2) requires repeated assembly of the system involving the basis matrix V, drastically compromising the performance. The Empirical Interpolation Method and its variants [49,50] are affine-recovery techniques which interpolate the non-affine or nonlinear terms, suggesting a considerable speed-up. Such methodologies entail an intrusive approach, requiring the a priori knowledge of the physics and its high-fidelity operators.

2.2. *Non-intrusive and data-driven approaches*

As detailed above, the intrusive nature of projection-based model order reduction can jeopardize the goal of traditional reduced order models, i.e. high computational efficiency. Non-intrusive and data-driven ROMs arise as an alternative to these techniques, with the scope of reducing the computational complexity without the need to project the governing equation. The starting point to build such ROMs is the dataset constituted by the aforementioned snapshots. Several linear and nonlinear methodologies have been developed and analyzed, occasionally benefiting by machine learning approaches.

The key reference of the first class is represented by the RB method, expressing the reduced approximation as the linear expansion over the basis functions as $u_{\mathcal{N}}(\boldsymbol{\mu}) \approx u_N(\boldsymbol{\mu}) = \mathsf{V}\mathbf{u}_N(\boldsymbol{\mu})$. While traditional methods require the solution of a reduced (nonlinear) system to obtain the reduced coefficient vector $\mathbf{u}_N(\boldsymbol{\mu})$, non-intrusive approaches exploit different workarounds to replace this step.

Proper Orthogonal Decomposition with interpolation (PODI) [51,52] projects the snapshots onto the POD basis and then interpolates the reduced coefficients. When interested in regression rather than interpolation, machine learning techniques become relevant.

With POD-NN, one recovers these by exploiting a feedforward neural network for a supervised learning task, where the dataset is given by the projected snapshots [10,11,53]. Physics-reinforced neural network (PRNN) addresses the low-data regime, augmenting the POD-NN loss with the reduced equations encoding the physics [54].

The aforementioned methodologies help to recover an efficient reconstruction of the solution, but they are still exploiting a linear basis expansion. Unfortunately, many problems cannot be effectively tackled by linear ROMs, such as (i) advection-dominated problems for which a large number of modes is required [55], and (ii) bifurcation problems, innately of nonlinear and ill-posed nature [56].

Nonlinear reduction techniques have been developed to cope with these issues. These involve an approximation of the form $\mathbf{u}_{\mathcal{N}}(\boldsymbol{\mu}) \approx \psi(\mathbf{u}_N(\boldsymbol{\mu}))$, where ψ is a suitable nonlinear map. The linear approach corresponds to $\psi(\mathbf{u}_N(\boldsymbol{\mu})) = \mathsf{V}\mathbf{u}_N(\boldsymbol{\mu})$. As alternatives, we highlight: the registration method [57], kernel principal component analysis [58], the method of freezing [59], the shifted POD [60], localized models [61] and the use of the Wasserstein space [62].

These methodologies allow to obtain much better accuracy, but are usually characterized by a less trivial online phase. This is where machine learning techniques come into play. As discussed in the introduction, the autoencoder architecture, with its nonlinear compression and decoding phases, represents the ideal generalization of the POD technique.

Recently, they have been widely used to develop nonlinear reduced order models. In [5], the authors present a nonlinear manifold least-squares Petrov-Galerkin method based on convolutional autoencoders. A hyper-reduced extension exploiting the physics has been introduced in [63]. The DL-ROM approach developed in [6] augments the training phase with a supervised task, in which the bottleneck is learned through a feedforward neural network. A POD preprocessing has been considered to reduce the size of the network in [64]. An interesting extension entails the reconstruction of the hyper-reduction operators [65]. Dynamic integration based on thermodynamics priors exploits autoencoders to learn mappings to a low-dimensional manifold [9,66]. Many other non-intrusive approaches have been proposed recently, e.g. based on Gaussian Process Regression (GPR) [67] and Operator Inference [68]. Finally, Neural Operators are successful in approximating mappings between function spaces, rather than functions themselves. These methodologies usually benefit from both the discretization-invariant and the universal approximation properties [69–71].

Following the path set by these works, graph convolutional autoencoders are the natural next step for the development of nonlinear model order reduction techniques, combining the classical PDE framework with solutions defined on unstructured grids and machine learning structures encoding geometric features.

3. Graph convolutional autoencoder for nonlinear model order reduction

Computational problems derived from physical models are naturally formulated over discretized domains for their analysis and resolution. In many cases, the geometry is not simple, suggesting the use of unstructured mesh. Deep learning techniques typically operate with structured datasets, e.g. standard CNNs, that depend strongly on the ordering of data points. Treating complex domain using such approaches often results in inefficient trainings, as they do not take the geometrical information into consideration.

Geometric Deep Learning arose as a mathematical framework to augment neural networks capabilities with geometric priors. This results in data driven approaches, characterized by both the spatial domain and the physical behavior of the system.

Graph Neural Networks are a branch of geometric deep learning. In this context, GNNs interpret the computational mesh as a peculiar graph structure, where the nodes are the vertices, and the edges constitute the boundary of each element of the triangulation. This representation makes the use of graph neural networks suitable for the development of computational problems defined on unstructured grids.

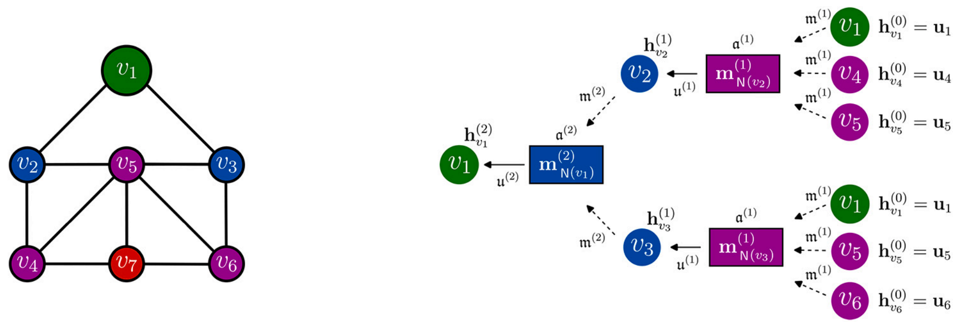
3.1. Brief introduction to graph neural networks

Let us consider the mesh $T(\mathcal{V}, \mathcal{E}, \mathcal{F})$, which can be seen as a simple, undirected, and connected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$ (see Fig. 2), including additional information about the elements of the mesh $\mathcal{F}$.

The nodes have associated features $\mathbf{u}$, that represent the evaluation of a set of state variables at the vertices of the mesh. This information can be embedded in the matrix $\mathbf{U} = \left[\mathbf{u}_1 | \dots | \mathbf{u}_{N_h}\right]^T \in \mathbb{R}^{N_h \times d}$, for all the N_h nodes of the graph,[1] which have no particular order, and d features, i.e. a scalar or vector field. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N_h \times N_h}$ records of the connections among nodes. A convenient way of expressing it is by means of the adjacency list, where the k-th entry, corresponding to $e_k \in \mathcal{E}$, is the pair (i, j) denoting the existence of a link between $v_i, v_j \in \mathcal{V}$. This is crucial to efficiently define operations which are permutation-invariant and equivariant. Thus, one can randomly permute the original ordering of the nodes induced by the mesh labeling without changing the final output. This is a key difference with standard CNNs, where a fixed filter produces different outputs if two pixels are swapped.

Starting from the graph structure as defined above, one obtains a graph neural network by defining a set of optimizable operations acting on all attributes of the graph. Despite their recent introduction, many works have focused on how to process graph-data through a neural network. Thus, let us review in more detail the basic operations which constitutes the backbone of many GNN architectures: (i) message passing framework, (ii) convolutional layers, and (iii) down-sampling and up-sampling procedures.

[1] The dataset is obtained by extracting the values of the high-fidelity solutions at the corresponding vertexes of the mesh, regardless the polynomial order of the Lagrange elements.



v_1
v_2
v_5
v_3
v_4
v_7
v_6
$\mathbf{h}^{(2)}_{v_1}$
$\mathfrak{u}^{(2)}$
$\mathfrak{a}^{(2)}$
$\mathbf{m}^{(2)}_{\mathsf{N}(v_1)}$
$\mathfrak{m}^{(2)}$
$\mathbf{h}^{(1)}_{v_2}$
$\mathfrak{u}^{(1)}$
$\mathfrak{a}^{(1)}$
$\mathbf{m}^{(1)}_{\mathsf{N}(v_2)}$
$\mathbf{h}^{(1)}_{v_3}$
$\mathbf{m}^{(1)}_{\mathsf{N}(v_3)}$
$\mathfrak{m}^{(1)}$
$\mathbf{h}^{(0)}_{v_1} = \mathbf{u}_1$
$\mathbf{h}^{(0)}_{v_4} = \mathbf{u}_4$
$\mathbf{h}^{(0)}_{v_5} = \mathbf{u}_5$
$\mathbf{h}^{(0)}_{v_6} = \mathbf{u}_6$

Fig. 2. Example of a graph with message passing and aggregation procedures, showing the flow of information to compute a node's hidden embedding, where colors denote the connectivity-based distance from node v_1 to be updated. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

3.1.1. *Message passing framework*

The main idea is to propagate the information to local neighborhoods of each node v, which is commonly denoted by $\mathsf{N}(v)$ and defines its computation graph with degree $|\mathsf{N}(v)|$. Such messages are exchanged between nodes at different k-hops (or layers) of the graph, and are updated by exploiting neural networks. At the k-th hop, one computes the hidden embedding $\mathbf{h}^{(k)}_u \in \mathbb{R}^{d(k)}$ of the node u, a vector representing a transformation of its original features by means of differentiable aggregate and update computations. In practice, at a node $u \in \mathcal{V}$, one assembles the messages to be sent through the operation $\mathfrak{m}^{(k)}$ as

$$\mathbf{m}^{(k)}_v = \mathfrak{m}^{(k)}(\mathbf{h}^{(k-1)}_v), \quad \text{from each node } v \in \mathsf{N}(u)$$

aggregates them with $\mathfrak{a}^{(k)}$ in

$$\mathbf{m}^{(k)}_{\mathsf{N}(u)} = \mathfrak{a}^{(k)}(\{\mathbf{m}^{(k)}_v,\ \forall v \in \mathsf{N}(u)\}),$$

and finally updates the hidden embedding by means of the function $\mathfrak{u}^{(k)}$

$$\mathbf{h}^{(k)}_u = \mathfrak{u}^{(k)}(\mathbf{m}^{(k)}_{\mathsf{N}(u)}).$$

For each node $v_j \in \mathcal{V}$, the initialization of the hidden embedding is defined as its own input features, i.e. $\mathbf{h}^{(0)}_{v_j} = \mathbf{u}_j$. Since we seek to update the hidden embedding without erasing previous information, we consider ghost self-edges, such that $\mathsf{N}(u)$ contains the node u itself.

The whole procedure can be performed in several ways. The simplest example of message function $\mathfrak{m}^{(k)}$ is the multiplication of the hidden embedding with a weight matrix $\mathbf{W}^{(k)} \in \mathbb{R}^{d(k)\times d(k-1)}$, i.e. $\mathbf{m}^{(k)}_v = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}_v$. For the aggregate function $\mathfrak{a}^{(k)}$, one usually considers the normalized sum of the neighbor embeddings given by $\mathbf{m}^{(k)}_{\mathsf{N}(u)} = \sum_{v\in\mathsf{N}(u)} \frac{\mathbf{m}^{(k)}_v}{|\mathsf{N}(u)|}$, to better balance nodes with much higher degree. Finally, the update operation $\mathfrak{u}^{(k)}$ can be seen as the nonlinear activation function $\mathbf{h}^{(k)}_u = \sigma(\mathbf{m}^{(k)}_{\mathsf{N}(u)})$ in the neural network context, e.g. with $\sigma(x) = \mathrm{ReLU}(x)$ or $\sigma(x) = \tanh(x)$.

To summarize, a basic GNN with K layers can be described by the iteration to update the node embeddings[2]

$$\mathbf{h}^{(k)}_u = \sigma\left(\frac{1}{|\mathsf{N}(u)|}\sum_{v\in\mathsf{N}(u)} \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}_v\right) \quad \text{for } k = 1,\dots,K, \text{ and } u \in \mathcal{V}. \tag{3}$$

A bias term $\mathbf{b} \in \mathbb{R}^{d(k)}$, omitted here for simplicity, improves the performance. Mimicking the standard neural networks' framework, one can also rewrite the message passing iteration (3) with the following compact graph-level notation

$$\mathbf{H}^{(k)} = \sigma\left(\mathbf{D}^{-1}(\mathbf{A}+\mathbf{I})\mathbf{H}^{(k-1)}\mathbf{W}^{(k)^T}\right) \quad \text{for } k = 1,\dots,K, \tag{4}$$

where $\mathbf{H}^{(k)} = \left[\mathbf{h}^{(k)}_{v_1}|\dots|\mathbf{h}^{(k)}_{v_{N_h}}\right]^T \in \mathbb{R}^{N_h\times d(k)}$ expresses the matrix with the hidden embedding for each node taken as row, $\mathbf{D}^{-1}$ is the diagonal matrix with the inverse of the degree of each node $|\mathsf{N}(v)|$, and $\mathbf{A}, \mathbf{I}$ are the adjacency and the identity matrices (representing the self-edges), respectively.

These operations are permutation invariant/equivariant, thus allowing for a message passing framework to produce the same results independent of the initial ordering of the nodes, and based solely on their connections.

[2] We remark that with the self-edges notation, in this explanatory example one cannot distinguish between information coming from the node itself and its neighbors. Sharing the same weight matrix $\mathbf{W}^{(k)}$ possibly compromising the capability of the GNN. A common approach is to consider different weight matrices for the two types of information.

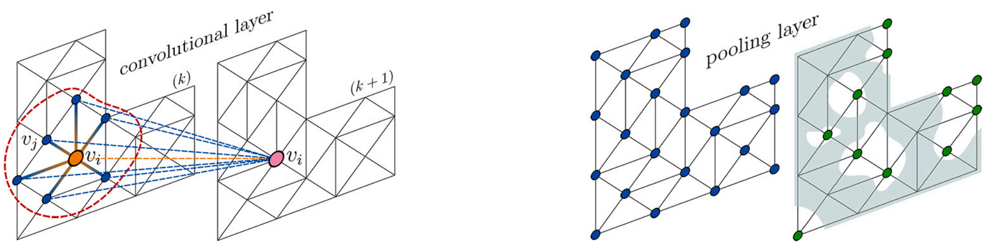


Fig. 3. Convolutional and pooling layers.

3.1.2. Convolutional layers in the non-Euclidean setting

A graph convolutional network (GCN) is a specific type of GNN aiming at extending the concept of CNNs, operating in regular Euclidean domains, to handle non-grid data. The GCN learns to combine the hidden embeddings by defining convolutional operations able to capture the relationships between the nodes, and optimize a given loss function (see Fig. 3). This process is similar to the standard convolutional layers in CNNs, where a fixed filter slides over the pixels of an image to produce gathered information. The main issue with CNNs is that the operations are not invariant w.r.t. the nodes' order. Given also the different cardinality of each node's neighbors, such extension to graphs is not trivial [23,40].

One can distinguish between two classes of convolutional layers: spectral and spatial ones. Spectral approaches are characterized by the spectral analysis of the graph Laplacian matrix to obtain a suitable basis. These methods have strong mathematical foundations from signal processing, but they depend on the graph's cardinality $|N_h|$, requiring an eigen-decomposition to define the filters, which compromises their efficiency. Such procedures define a global filter that can be applied to the whole graph.

Spectral convolutional methods seek to build the filters as $g_{\mathbf{w}} * \mathbf{H} \doteq \mathbf{U} g_{\mathbf{w}} \mathbf{U}^T \mathbf{H}$, where $g_{\mathbf{w}}$ is the filter, and $\mathbf{U}$ the eigenvector matrix of the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$. While ChebNet exploits a Chebyshev polynomial of degree K in $\mathbf{L}$ to approximate the convolution $g_{\mathbf{W}} * \mathbf{H}$, GCN simplifies this by considering only the first term [23].

Conversely, spatial convolutions reflect an ordinary CNN approach, where each node is related to its local neighbors. Being local rather than global, they do not depend on the graph's dimensionality, resulting in faster and more efficient methods.

Spatial methods are mainly based on the message-passing operation, which can be defined in several ways. In GraphSage [72] features are sampled and then aggregated, while in GINN [73] the adjacency matrix is exploited for regression tasks. In this work, we have decided to exploit the MoNet spatial framework introduced in [24], which can be interpreted as a Gaussian Mixture Model (GMM), and thus a general class for convolutions in non-Euclidean domains. MoNet builds a set of pseudo-coordinates $\mathbf{e}$ used to define the weights of an optimizable Gaussian kernel with Q filters, through the iteration procedure

$$\mathbf{h}_u = \frac{1}{|\mathsf{N}(u)|} \sum_{v \in \mathsf{N}(u)} \frac{1}{Q} \sum_{q=1}^{Q} \boldsymbol{\omega}^q(\mathbf{e}_u) \odot \mathbf{W}^q \mathbf{h}_v, \tag{5}$$

where $\odot$ is the element-wise multiplication, and $\boldsymbol{\omega}^q$ is the weighting function defined in terms of a trainable mean vector $\boldsymbol{\mu}_q$ and a diagonal covariance matrix $\boldsymbol{\Sigma}_q$ as

$$\boldsymbol{\omega}^q(\mathbf{e}_u) = \exp\left(-\frac{1}{2}(\mathbf{e}_u - \boldsymbol{\mu}_q)^T \boldsymbol{\Sigma}_q^{-1} (\mathbf{e}_u - \boldsymbol{\mu}_q)\right).$$

In practice, MoNet considers as pseudo-coordinate the edge attributes given by the distance between two connected nodes, introducing a geometric bias in the learning process. These geometrically-informed convolutions could increase the cost of the learning task w.r.t. other convolutional layers, but they also improve the generalizability.

To conclude, we recall that GCN can be seen as a spatial method, and thus a particular instance of MoNet with $Q = 1$ and $\boldsymbol{\omega}^1 = 1/|\mathsf{N}(u)|$. In this more general setting the filters $\boldsymbol{\omega}^q$ are optimizable.

3.1.3. Down-sampling and up-sampling procedures

Another key difference between CNNs and GNNs consists in the property of the former to naturally reduce the spatial dimension of the input. As we have seen in the previous section, the convolutional layers are defined for each node of the original graph/mesh. For this reason, in some cases, it can be useful to rely on down-sampling and up-sampling (see Fig. 3). These methodologies are needed in the encoder and the decoder, to obtain representations of the graph which are, respectively, coarser and finer.

Pooling is the most widely used technique to down-sample the size of the input by aggregating information from multiple nodes and edges. This results in a smaller and more manageable graph, improving generalization and performance. Although it is a well-defined operation for CNNs, it cannot be directly applied to graphs, as there is no natural hierarchy in nodes' importance. The selection of subsets of nodes can be performed in several ways, from random mask selection to more sophisticated operations including clustering and attention mechanisms [39,74,75]. Applying down-sampling techniques multiple times in a GNN, one can construct a multiscale hierarchy of coarser domains in the spirit of algebraic multigrid methods [76]. However, when misused, these approaches result in information loss, and a deteriorating model performance.

Un-pooling can be considered the inverse operation of pooling and belongs to the class of up-sampling techniques. While it is relatively simple to collect information to reduce the number of nodes, the reverse is still an open research challenge. Depending

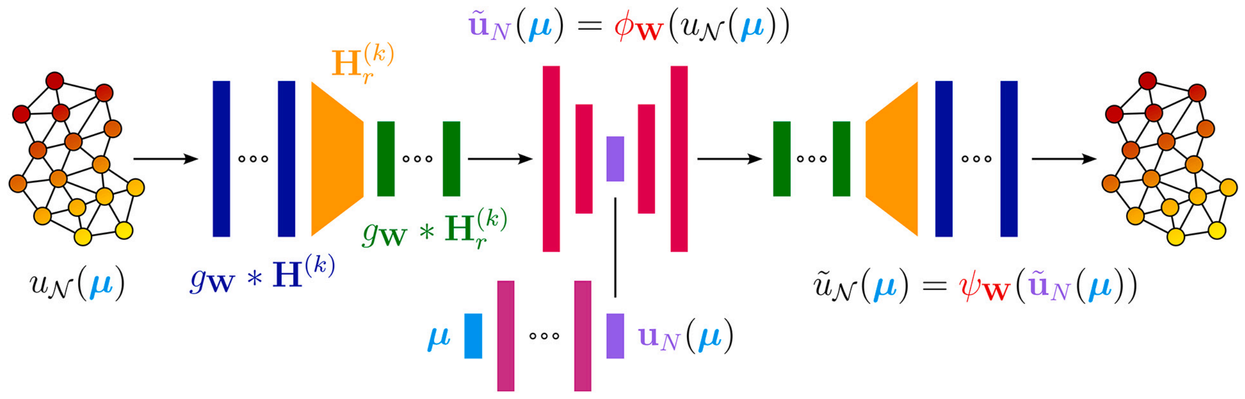

$\tilde{\mathbf{u}}_N(\boldsymbol{\mu}) = \phi_{\mathbf{W}}(u_{\mathcal{N}}(\boldsymbol{\mu}))$
$\mathbf{H}_r^{(k)}$
$u_{\mathcal{N}}(\boldsymbol{\mu})$
$g_{\mathbf{W}} * \mathbf{H}^{(k)}$
$g_{\mathbf{W}} * \mathbf{H}_r^{(k)}$
$\tilde{u}_{\mathcal{N}}(\boldsymbol{\mu}) = \psi_{\mathbf{W}}(\tilde{\mathbf{u}}_N(\boldsymbol{\mu}))$
$\boldsymbol{\mu}$
$\mathbf{u}_N(\boldsymbol{\mu})$

Fig. 4. GCA-ROM architecture for the offline phase.

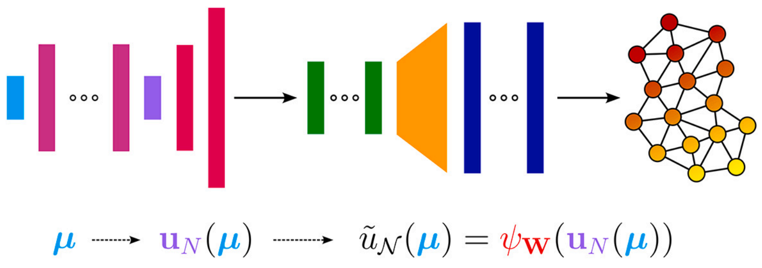

$\boldsymbol{\mu}$
$\mathbf{u}_N(\boldsymbol{\mu})$
$\tilde{u}_{\mathcal{N}}(\boldsymbol{\mu}) = \psi_{\mathbf{W}}(\mathbf{u}_N(\boldsymbol{\mu}))$

Fig. 5. GCA-ROM architecture for the online phase.

on the ML application, up-scaling can be achieved by storing the geometric information obtained during the coarsening step, e.g. for classification, segmentation, or noise filtering purposes. COMA stores the barycentric information of the eliminated points [39], while Mincut requires the adjacency matrix [74]. To our knowledge, there are only a few functions that perform up-sampling without information obtained during the down-sampling step. This is crucial to obtain an online reconstruction fully decoupled from the original solution field, i.e. encoder-free. PointNet++ proposes a k-NN interpolation of the points to up-sample by considering the position and the features of the nodes in the down-sampled coarser configuration [46]. In particular, given a node at position $\mathbf{x}_i$, we define its feature vector $\mathbf{u}_i$ as the weighted interpolation w.r.t. its k neighbors given by

$$\mathbf{u}_i = \frac{\sum_{j=1}^{k} \xi(\mathbf{x}_j)\mathbf{u}_j}{\sum_{j=1}^{k} \xi(\mathbf{x}_j)}, \qquad \text{where} \quad \xi(\mathbf{x}_j) = \frac{1}{d(\mathbf{x}_i, \mathbf{x}_j)^2}.$$

We remark that, in this preliminary investigation, the goal of the pooling procedures is restricted to the study of the learning capabilities from a coarser representation of the mesh, obtained e.g. from a fixed set of sensors. In the following we will show that, if the loss of information is not excessive, our naive approach is able to recover the field with acceptable accuracy. This difficult task is of crucial importance to drastically reduce the number of trainable parameters and speed-up the convergence. Performing an optimal pooling remains open, with interesting works about multiscale layers [77,78], and will be the subject of future investigations.

3.2. A graph convolutional autoencoder approach for reduced order modeling

We are now ready to propose the graph convolutional autoencoder for model order reduction applications. The main sources of inspiration are CNN-based autoencoder architecture introduced in [6,5]. As detailed in the previous sections, such approaches are particularly useful when dealing with structured meshes, that can be seen as images with a fixed number of neighboring pixels. For this reason, we consider a similar architecture, but extend its applicability in a geometrically consistent manner to unstructured meshes defined over complex domains, when a Cartesian representation is no longer possible.

ML approaches mimic the standard ROM setting, having an offline phase in which one forms the dataset and build the reduced model by training the neural network, and an online phase for the real-time evaluation of the field of interest.

We consider the graph dataset $\Xi = \{\mathbf{u}_{\mathcal{N}}(\boldsymbol{\mu}^i), \Omega_{\mathcal{N}}(\boldsymbol{\mu}^i)\}_{i=1}^{N_S}$, formed by N_S solutions $\mathbf{u}_{\mathcal{N}}(\boldsymbol{\mu}^i)$ of a parametrized PDE defined over unstructured meshes $\Omega_{\mathcal{N}}(\boldsymbol{\mu}^i)$, corresponding to the set of parameters $\{\boldsymbol{\mu}^i\}_{i=1}^{N_S}$. Following the first DL-ROM approach [6], we detail the steps of the modular architecture and refer to Figs. 4 and 5, respectively, for the offline and online phases.

The architecture for the offline training is composed of an autoencoder and a multi-layer perceptron (MLP). The former seeks to approximate the identity map while encoding the information into a low-dimensional space expressed by the bottleneck or latent space.

We feed the encoder module $\phi_{\mathbf{W}}$ with the graph data Ξ, exploiting MoNet (5) as the message passing algorithm. In this step, by means of graph convolutions $g_{\mathbf{W}} * \mathbf{H}^{(k)}$, we extract the most meaningful information between the nodes and their evolution w.r.t. the samples. When interested in reducing the graph dimensionality, one can consider an additional module by applying a fixed random mask for down-sampling, to obtain a reduced graph with $\mathbf{H}_r^{(k)}$ hidden embeddings. A second batch of convolutional layers $g_{\mathbf{W}} * \mathbf{H}_r^{(k)}$ can be used to further encode the latent space. To improve training, we also augment the network with a residual learning strategy exploiting skip-connections, featuring identity shortcuts inspired by the ResNet framework developed in [79]. This step, which can only be applied in batches of convolutions sharing the same graph's size, enables the architecture to acquire the residual knowledge between layers, while simultaneously promoting the flow of gradient information throughout the network. At this stage, we use fully-connected layers to connect the graph structure with the bottleneck. We thus encode the original information in the latent vector $\tilde{\mathbf{u}}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$, obtained as the evaluation of the solution field through the encoder structure $\tilde{\mathbf{u}}_N(\boldsymbol{\mu}) = \phi_{\mathbf{W}}(\mathbf{u}_{\mathcal{N}}(\boldsymbol{\mu}))$. The decoding structure is composed by the same operations as the encoding ones, but in reversed order. This way, processing the bottleneck with the decoder $\psi_{\mathbf{W}}$, we obtain the reconstruction of the original field $\tilde{u}_{\mathcal{N}}(\boldsymbol{\mu}) = \psi_{\mathbf{W}}(\tilde{\mathbf{u}}_N(\boldsymbol{\mu}))$, and we can assemble the first term in the loss function for the unsupervised learning task as

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N_{\mathrm{tr}}} \sum_{i=1}^{N_{\mathrm{tr}}} \left\| \tilde{\mathbf{u}}_{\mathcal{N}}(\boldsymbol{\mu}^i) - \mathbf{u}_{\mathcal{N}}(\boldsymbol{\mu}^i) \right\|_2^2 .$$

Having encoded the latent variables with the autoencoder, we use this dataset for the supervised learning task with the MLP. Thus, we are non-intrusively building the map $\mathbf{u}_N(\boldsymbol{\mu}) = \mathrm{MLP}(\boldsymbol{\mu})$, from which we can recover the reduced coefficient. The MLP, fundamental for the online evaluation phase, defines the loss

$$\mathcal{L}_{\mathrm{BTT}} = \frac{1}{N_{\mathrm{tr}}} \sum_{i=1}^{N_{\mathrm{tr}}} \left\| \mathbf{u}_N(\boldsymbol{\mu}^i) - \tilde{\mathbf{u}}_N(\boldsymbol{\mu}^i) \right\|_2^2 .$$

These two Mean Squared Error (MSE) loss terms can be balanced through the hyperparameter λ, resulting in the loss function

$$\mathcal{L} = \mathcal{L}_{\mathrm{MSE}} + \lambda \mathcal{L}_{\mathrm{BTT}},$$

which guides the training procedure through the Adam optimizer. Finally, during the online phase, the bottleneck can be directly evaluated by exploiting the MLP for a new parameter $\boldsymbol{\mu}$, and successively decompressed through the graph decoder, to recover the corresponding field defined over its geometry.

As for the preprocessing of the dataset, we applied a double normalization with respect to the sample and the features, i.e. we impose zero mean and unitary variance for each column (i.e. node-wise), and subsequently for each row (i.e. parameter-wise). This allowed us to treat all the benchmarks within a unique framework, greatly improving the results. Moreover, we split the dataset Ξ into training and testing set w.r.t. the percentage rate r_{t} in the following manner: $\Xi_{\mathrm{tr}} = \mathsf{shuffled}(\Xi)[:, :r_{\mathrm{t}} N_{\mathrm{S}}]$ and $\Xi_{\mathrm{te}} = \Xi \setminus \Xi_{\mathrm{tr}}$ of dimension N_{te}.

Common disadvantages of machine learning approaches are the tuning of the hyperparameter and the learning procedure. As we will show in Section 4, the GCA-ROM architecture is quite robust w.r.t. both. Among the possible quantities to be investigated we focus on the following set: the training rate percentage r_{t}, the pooling rate percentage r_{p}, the number of nodes in the feedforward network of the GCA ffn, the number of nodes in each MLP layer of the latent map n_l, the dimension of the bottleneck n, the weight λ in the loss function, and the number of a-priori and a-posteriori hidden channels, hcp and hcd, respectively.

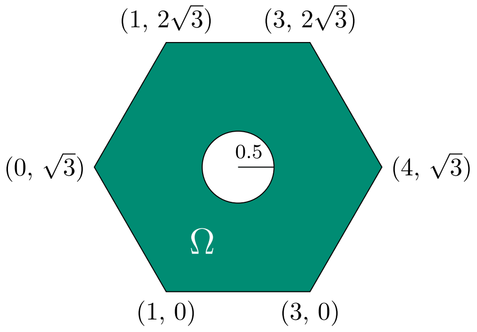
The scope of the hyperparameter analysis (showed in Appendix A.1) is not to produce the best possible accuracy for the model, but to explore the sensitivity of the architecture to different autoencoder configurations. For this reason, we have not focused on the usual ones, e.g. activation functions, learning rates and number of epochs, nor exploited multiple restart approaches with different seeds. In particular, we considered the combinations of the following configurations for the aforementioned hyperparameters[3]: $r_{\mathrm{t}} \in [10, 30, 50]$, $r_{\mathrm{p}} \in [30, 50, 70]$, $\mathrm{ffn} \in [100, 200, 300]$, $n_l \in [50, 100]$, $n \in [15, 25]$, $\lambda \in [0.1, 1, 10]$, and $\mathrm{hcp}, \mathrm{hcd} \in [1, 2, 3]$.

We remark that, as in CNNs, our GNN architecture shares the weights across the nodes,[4] making it scalable to large datasets, in contrast to fully-connected autoencoders. This results in a much lower number of trainable parameters, and substantial speed-up in the training phase. The size of the optimization problem is thus dominated by the MLP dimension. For this reason the pooling approach could be fundamental for finer meshes.

This geometry-informed architecture not only reduces the computational complexity, but also fosters learning. Indeed, already with a relatively small training horizon, i.e. $N_{\mathrm{epochs}} = 5000$, the network reaches accurate reconstruction errors for the benchmarks, and still show a decreasing behavior. Finally, we note that GCA-ROM is a convolve-then-reshape method, extracting the inductive bias directly from its original framework, while the ones exploiting standard CNNs are reshape-then-convolve approaches.

[3] In the pooling case, testing the architecture without convolutional layers is equivalent to learning the map only from sensors data.

[4] The weights multiply the hidden embedding $\mathbf{h}_u^{(k)}$, thus the matrix $\mathbf{W}^{(k)}$ depends only on the number of features and filters, but not on the number of neighbors $|\mathsf{N}(u)|$. This allows for many possible generalizations, from different meshes to unseen nodes.



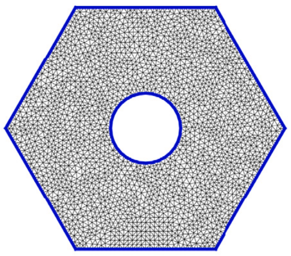


Fig. 6. The hexagonal domain with a hole for the nonlinear elliptic problem and its computational mesh.

4. Numerical results

In this section, we present different applications showcasing the capabilities of GCA-ROM as an efficient nonlinear reduced order model. The main goal of this approach consists in providing an interpretable and effective framework to investigate parametric PDEs defined over parametrized domains, to enhance the performance of linear ROMs, and to decrease the number of expensive simulations during the offline phase. Toward these goals, we highlight the main properties of the GCA-ROM architecture when facing these issues: (i) its consistency and versatility when dealing with unstructured varying geometries, (ii) its strength in providing better accuracy/speedup for problems with slow Kolmogorov n-width decay, and (iii) its expressive power, characterized by high generalizability even in the low-data regime.

We analyze the performance of the proposed methodology with respect to four benchmark parametrized PDEs: (i) nonlinear Poisson equation, (ii) advection-dominated problem, (iii) Graetz flow, and (iv) Navier-Stokes system. Each of these problems features a different complexity that GCA-ROM is capable of handling. We compare the performance of the novel architecture w.r.t. standard intrusive techniques such as POD-Galerkin method, and the original data-driven DL-ROM, both in terms of accuracy and speed-up. As detailed below, the method presents remarkable improvements w.r.t. state-of-the-art methodologies, and paves the way for new investigations bridging standard numerical techniques for PDEs and neural network approaches.

In particular, we test the architecture, with and without exploiting up- and down-sampling operations. This way, one can distinguish between the generalization property in the plain setting, and the case in which one has only partial access to the high-fidelity solutions, i.e. through sensors.

As concerns the error analysis, we denote with

$$\varepsilon_{GCA}(\boldsymbol{\mu}) = \frac{\left\|\mathbf{u}_{\mathcal{N}}(\boldsymbol{\mu}) - \tilde{\mathbf{u}}_{\mathcal{N}}(\boldsymbol{\mu})\right\|_2}{\left\|\mathbf{u}_{\mathcal{N}}(\boldsymbol{\mu})\right\|_2} \quad \text{and} \quad \bar{\varepsilon}_{GCA} = \frac{1}{(1-r_{\mathrm{t}})N_{\mathrm{S}}} \sum_{\boldsymbol{\mu}\in\Xi_{\mathrm{te}}} \varepsilon_{GCA}(\boldsymbol{\mu})$$

the GCA-ROM relative error for a given parameter $\boldsymbol{\mu} \in \mathcal{P}$ and its mean over the testing set Ξ_{te}, respectively.

Notice that, rather than finding the optimal configuration of the architecture, we focus on testing several hyperparameter's configurations to investigate the robustness of the methodology w.r.t. the training procedure and the influence of the graphs' architecture. Thus, we will show the best results within our analysis, and refer to Appendix A for a more detailed description of the test cases and architectures, and performance w.r.t. different configurations.

4.1. Nonlinear Poisson equation

The first benchmark consists of a nonlinear elliptic problem in a two-dimensional hexagonal domain Ω with a hole, depicted in Fig. 6. With this toy problem, we aim at testing the ability of GCA-ROM to cope with an unstructured mesh with $N_h = 2562$ nodes, where the Euclidean geometry based on Cartesian coordinates does not provide a valid description of the domain.

The continuous formulation of the PDE governing this parametrized problem reads as: given $\boldsymbol{\mu} \in \mathbb{P}$, find $u(\boldsymbol{\mu})$ such that

$$-\nabla^2 u(\boldsymbol{\mu}) + \mu_1 \frac{e^{\mu_2 u(\boldsymbol{\mu})} - 1}{\mu_2} = g(\boldsymbol{x}),$$

where the physical multi-parameter $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{P} = [0.01, 10]^2$ controls the strength of the sink term and of the nonlinearity, and the source term is characterized by the following expression

$$g(\boldsymbol{x}) = 100 \sin(2\pi x)\cos(2\pi y), \quad \forall \boldsymbol{x} = (x, y) \in \Omega.$$

The dataset is constituted by $N_{\mathrm{S}} = 100$ snapshots (10 equispaced parameter samples in each direction) by means of Finite Element approximations over a uniform grid in $\mathbb{P}$. The nonlinear term prevents the straightforward applicability of ROMs (hyper-reduction techniques are needed to recover the efficiency), while the lack of a structured mesh leads to a geometrically inconsistent use of standard CNNs.

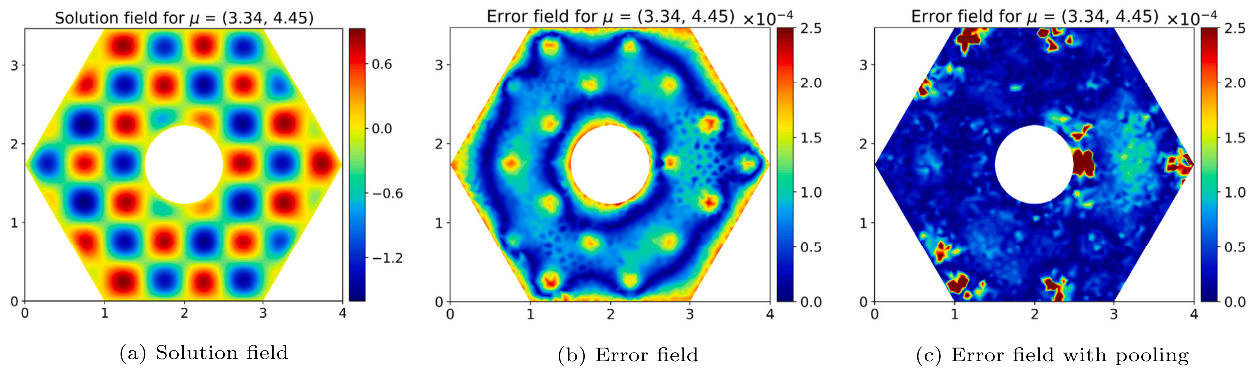


Fig. 7. Poisson problem for $\boldsymbol{\mu} = (3.34, 4.45)$.

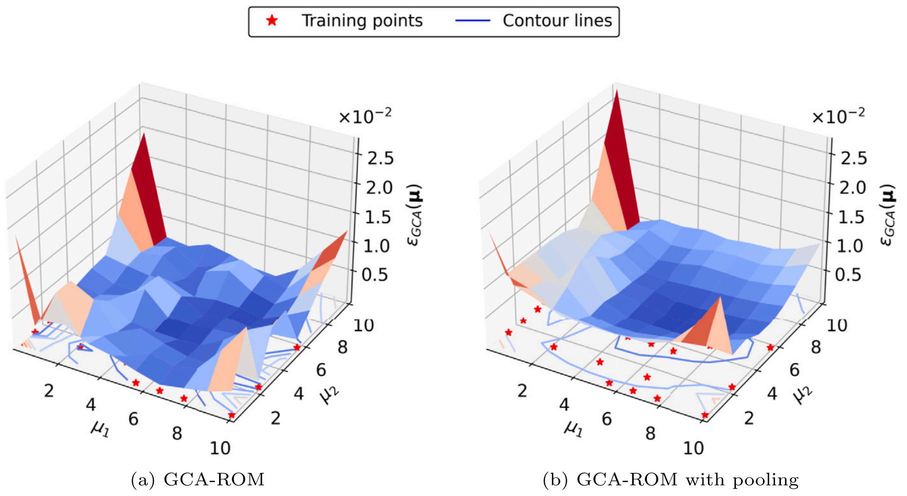


Fig. 8. GCA-ROM relative errors for the Poisson problem on the dataset Ξ, with red markers corresponding to the parameters used in the training set Ξ_{tr}.

We show in Fig. 7 the high-fidelity solution for Poisson obtained for $\boldsymbol{\mu} = (3.34, 4.45) \in \Xi_{\text{te}}$, and the GCA-ROM relative errors obtained with and without exploiting pooling procedures. In both cases, we can accurately reconstruct the corresponding field, with the pooling architecture performing slightly worse due to the high peaks in some regions of the domain, $\epsilon_{\text{GCA}}(\boldsymbol{\mu}) = 5.0 \times 10^{-3}$ and $\epsilon_{\text{GCA}}(\boldsymbol{\mu}) = 4.3 \times 10^{-3}$, respectively for pooling and plain versions. This is expected given the availability of only partial information over the solution field during the training stage.

In Fig. 7b, the error is higher near the maxima of the field, where the dataset Ξ shows the higher variance w.r.t. the sampling, and at the boundary of the domain. This is clearly an undesired effect of the convolution procedure, sharing the message with neighbors even where we impose homogeneous Dirichlet boundary conditions. However, this effect is only present for a few instances of the parameter, while in general the error is spatially distributed in the whole domain, confirming that, regardless the complexity of the solution manifold, GCA-ROM learns the intrinsic behavior and generalize to unseen data. The augmentation of our data-driven architecture with physical information can fix such behavior by guiding the learning task towards physically consistent fields, and will be the subject of forthcoming investigations. With the pooling approach, see Fig. 7c, the accuracy drops especially near the down-sampled regions, where we apply the mask.

To highlight the generalization capabilities achieved by GCA-ROM, we show in Fig. 8, for both approaches, the evolution of the relative error $\epsilon_{GCA}(\boldsymbol{\mu})$ in the whole dataset Ξ. As expected, the accuracy of the pooling approach, both in terms of mean and maximum errors, marginally deteriorates due to the lack of information, but still produces remarkably close results. For the plain architecture in Fig. 8a, it can be seen that, exploiting only $r_{\text{t}} = 30\%$ of the original dataset (reported in the plot with red stars), the relative error over Ξ_{te} is below 2×10^{-2} and its mean is $\bar{\epsilon}_{GCA} = 8 \times 10^{-3}$. The accuracy of the pooling procedure in Fig. 8b is also surprisingly good, again exploiting only $r_{\text{t}} = 30\%$ of the dataset, but also retaining only $r_{\text{p}} = 70\%$ of each high-fidelity solution, i.e. we do not have access to these fixed nodes. We are still able to reconstruct the Poisson solutions with maximum relative error below 2.7×10^{-2} and mean $\bar{\epsilon}_{GCA} = 1 \times 10^{-2}$.

This result shows the potential of graph convolutional architecture in capturing parametric behavior, even when dealing with sparse measurements, and paves the way for the development of robust up- and down-sampling procedures.

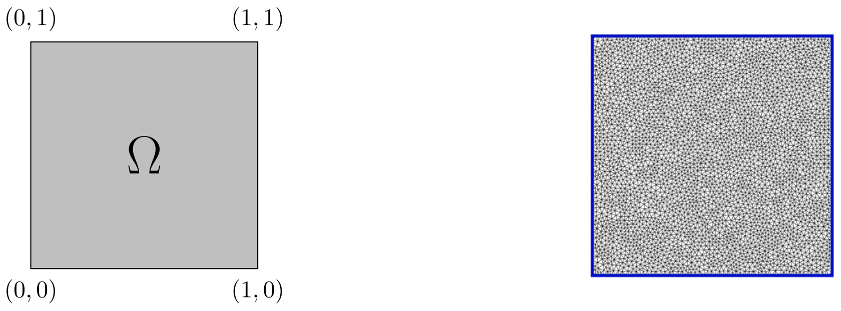


Fig. 9. The square domain for the advection-dominated problem and its computational mesh.

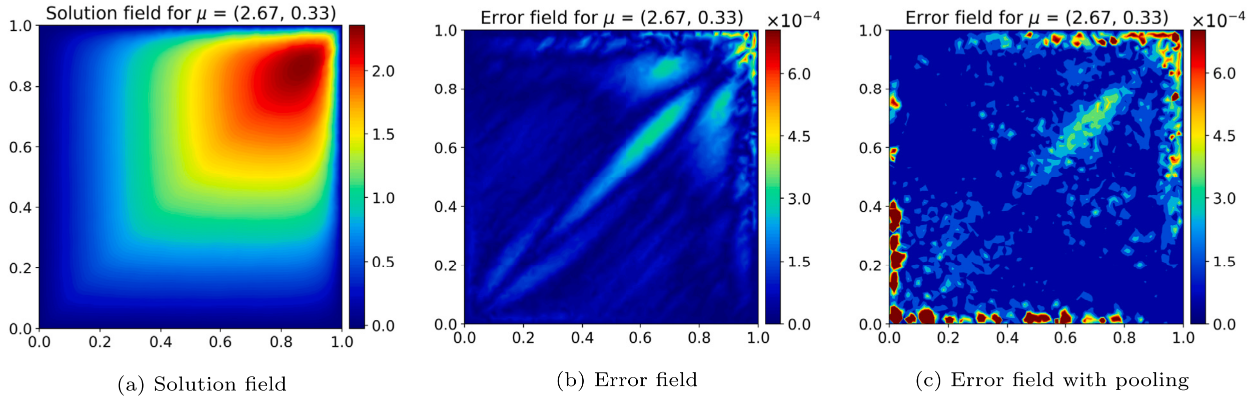


Fig. 10. Advection problem for $\boldsymbol{\mu} = (2.67, 0.33)$.

4.2. Advection dominated problem

Standard projection-based reduced order models struggle with slow Kolmogorov decay phenomena, such as those in the advection-dominated regime. Here, we test the performance of our nonlinear and data-driven approach for an advection-diffusion equation with a parametrized transport term. The domain is given by the unstructured mesh defined over the square domain $\Omega = [0,1]^2 \in \mathbb{R}^2$, depicted in Fig. 9, with $N_h = 3967$ nodes.

The advection-dominated problem reads as: given $\boldsymbol{\mu} \in \mathbb{P}$, find $u(\boldsymbol{\mu})$ such that

$$-\alpha(\mu_1)\Delta u(\boldsymbol{\mu}) + \beta(\mu_2)\cdot\nabla u(\boldsymbol{\mu}) = g \quad \text{in } \Omega,$$

where $\alpha(\mu_1) = 10^{-\mu_1}$ represents the diffusion coefficient defining the Péclet number, $\beta(\mu_2) = (\mu_2, \mu_2)$ is the direction of the transport, and $g = 1$ is the source term. Thus, the model is characterized by the physical multi-parameter $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{P} = [0,6]\times[-1,1]$. As before, we form the dataset by means of $N_S = 100$ snapshots over a uniform grid in $\mathbb{P}$.

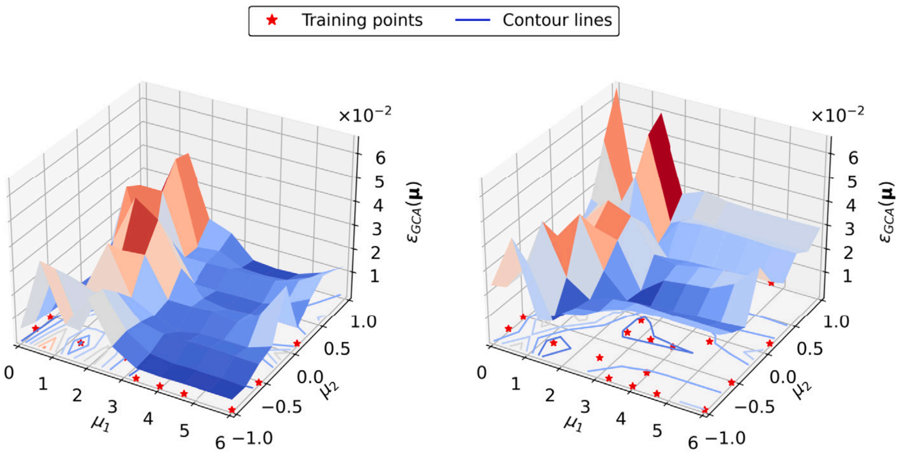
In Fig. 10 we plot the solution of the advection dominated problem for $\boldsymbol{\mu} = (2.67, 0.33) \in \Xi_{\mathrm{te}}$, and the two GCA-ROM relative error fields. Again, the qualitative behavior of the methodology shows accurate reconstructions, with the maximum errors located near the boundary layer's region. Given the different amount of information, the error in Fig. 10b is more distributed over the domain, and one order of magnitude smaller than the one in Fig. 10c.

For the accuracy, we plot in Fig. 11 the relative errors $\varepsilon_{GCA}(\boldsymbol{\mu})$ in the plain and down-sampled architectures over the dataset Ξ. In both cases, the training set comprises only $r_t = 30\%$ of the computed snapshots, but the maximum relative error over Ξ_{te} is around 5.2×10^{-2}, while its mean is $\bar{\varepsilon}_{GCA} = 2.4\times10^{-2}$. We stress the fact that to learn this complex behavior, we are only exploiting 30 high-fidelity solutions fixed a-priori by the seed, and testing on the remaining 70. The pooling strategy gives comparable results by selecting $r_p = 70\%$ nodes, and thus masking 30% of them, with maximum relative error below 7.2×10^{-2} and mean $\bar{\varepsilon}_{GCA} = 3.3\times10^{-2}$.

The graph structure is thus still able to predict unseen patterns, even when dealing with a combination of: a low-data regime, scattered mesh information, and complex parametric behavior compromising the linear reducibility of the model.

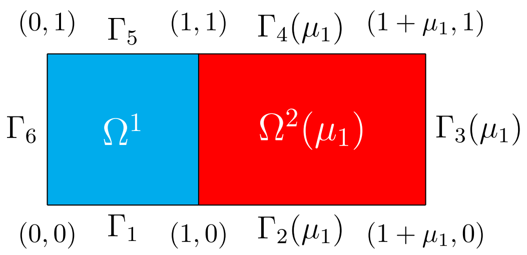
4.3. Graetz flow model

One of the main advantages of the GCA-ROM architecture is the possibility to work directly on the (unstructured) computational domain, introducing geometric priors and defining physically consistent convolutional operations. Now, we consider the case in



(a) GCA-ROM (b) GCA-ROM with pooling

Fig. 11. GCA-ROM relative errors for the Advection problem on the dataset Ξ, with red markers corresponding to the parameters used in the training set Ξ_{tr}.



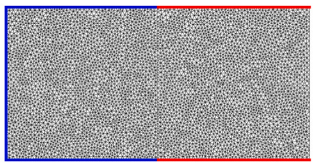


Fig. 12. The geometrically parametrized domain for the Graetz problem and its computational mesh.

which the PDE is posed on a geometrically parametrized domain,[5] i.e. when the distance between nodes changes w.r.t. the parameter. Standard convolutions are agnostic to this behavior. We study a steady-state Graetz problem on the parametrized rectangular geometry $\Omega(\mu_1) = \Omega_1 \cup \Omega_2(\mu_1) \in \mathbb{R}^2$ depicted in Fig. 12, with $\Omega_1 = [0,1] \times [0,1]$ and $\Omega_2 = [0,\mu_1] \times [0,1]$, and discretized with $N_h = 5160$ nodes.

The Graetz model describes a thermal field under the combined effects of heat's diffusion and advection along the horizontal direction, with homogeneous and non-homogeneous boundary conditions. The continuous PDE's formulation of the Graetz problem reads as: given $\boldsymbol{\mu} \in \mathbb{P}$, find $u(\boldsymbol{\mu})$ such that

$$\begin{cases} -\mu_2 \Delta u(\boldsymbol{\mu}) + y(1-y)\partial_x u(\boldsymbol{\mu}) = 0 & \text{in } \Omega(\mu_1), \\ u(\boldsymbol{\mu}) = 0 & \text{on } \Gamma_{\text{D}}, \\ u(\boldsymbol{\mu}) = 1 & \text{on } \Gamma_{\text{G}}(\mu_1), \\ \frac{\partial u}{\partial n}(\boldsymbol{\mu}) = 0 & \text{on } \Gamma_{\text{N}}(\mu_1), \end{cases}$$

where $\boldsymbol{n}$ denotes the outer normal to the boundary Γ_{N}. We impose homogeneous Dirichlet conditions on $\Gamma_{\text{D}} = \Gamma_1 \cup \Gamma_5 \cup \Gamma_6$, while the temperature is kept at $u = 1$ at the boundaries $\Gamma_{\text{G}} = \Gamma_2(\mu_1) \cup \Gamma_4(\mu_1)$. Finally, homogeneous Neumann conditions are imposed on $\Gamma_{\text{N}} = \Gamma_3(\mu_1)$.

The model has been parametrized by means of a physical and geometrical multi-parameter $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{P} = [1,3] \times [0.01, 0.1]$, controlling the length μ_1 of the idealized pipes, and the diffusivity coefficient μ_2. The dataset is formed by $N_{\text{S}} = 200$ snapshots, computed on a uniform grid in $\mathbb{P}$ with respectively 10 and 20 equispaced values for the two parameters.

In Fig. 13 we plot the solution of the Graetz model obtained for $\boldsymbol{\mu} = (2.78, 0.01) \in \Xi_{\text{te}}$, comparing the relative errors for two GCA-ROM approaches. The error in Fig. 13b has a maximum in the center of the stretched subdomain, where the transport phenomena take place, while the pooling approach in Fig. 13c produces higher errors at the interface between the two boundary conditions.

Fig. 14 shows the investigation of the reconstruction accuracy over the dataset Ξ through the relative errors $\varepsilon_{GCA}(\boldsymbol{\mu})$. The plain GCA-ROM architecture is still able to learn the low-dimensional evolution in the context of a varying geometry by exploiting only $r_{\text{t}} = 30\%$ of the dataset information, with the relative error over Ξ_{te} always below 6.7×10^{-2}, with mean $\bar{\varepsilon}_{GCA} = 5.6 \times 10^{-3}$. The

[5] The geometric parametrization is performed w.r.t. affine mappings, thus producing a set of snapshots defined on the same number of nodes as the original mesh cardinality.

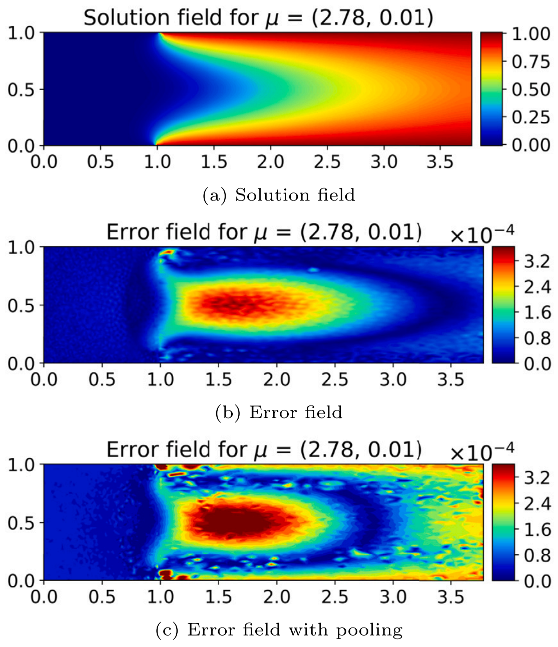


Fig. 13. Graetz flow model for $\boldsymbol{\mu} = (2.78, 0.01)$.

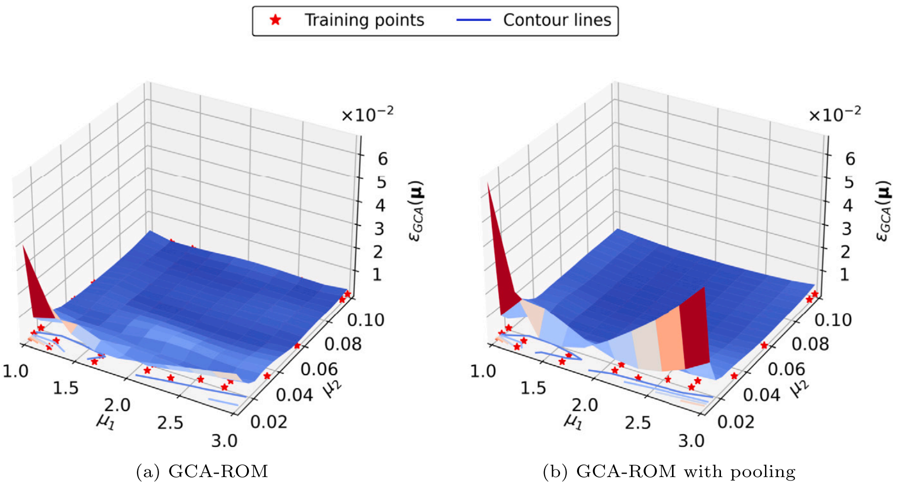


Fig. 14. GCA-ROM relative errors for the Graetz problem on the dataset Ξ, with red markers corresponding to the parameters used in the training set Ξ_{tr}.

results for the pooling approach are again comparable, with maximum relative error below 9.3×10^{-2} and mean $\bar{\varepsilon}_{GCA} = 7 \times 10^{-3}$, when exploiting $r_p = 70\%$ of the mesh. In both cases, we observe high generalization accuracy of the GCA-ROM in capturing the geometric parametrization. This is a result of the introduction of the geometric priors through the GNN architecture endowed with the MoNet convolutional layers. The higher errors are indeed localized for smaller values of the diffusivity parameter, where the phenomenon approaches the advection dominated regime.

4.4. Navier-Stokes bifurcating system

As the last benchmark, we consider a more complex problem that combines nonlinear terms, parametrized geometry and a vector unknown with non-unique solutions. The so-called Coanda's effect in the Navier-Stokes system is a typical example of fluid dynamic bifurcation problem [11,80–82]. For such problems, a small variation of the viscosity, in the neighborhood of the bifurcation points, produces a sudden change in the qualitative behavior of the solution, causing the co-existence of different states for the same parameter value. Here, we are not interested in recovering all the possible solutions. Rather, we seek to investigate the GCA-ROM performance in the approximation of the transition between two qualitatively different stable states, i.e. the symmetric flux and the wall-hugging profile.

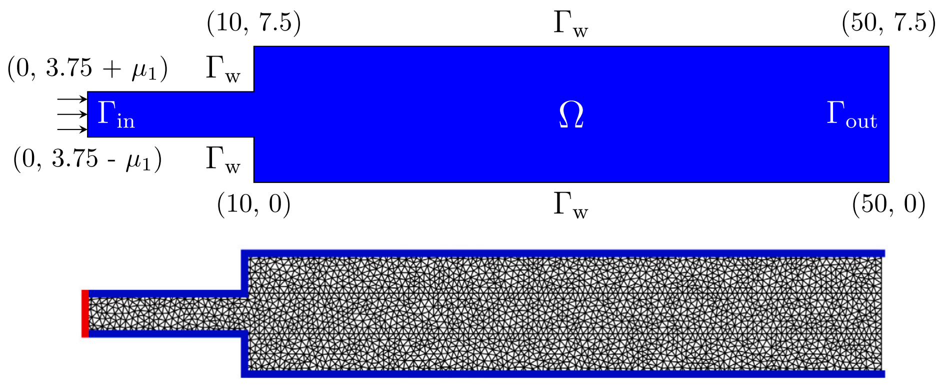


Fig. 15. The geometrically parametrized domain for the Coanda effect problem and its computational mesh.

Table 1
Configuration and results for the Navier Stokes benchmark.

Application	Sampling	Dataset		Component	$\bar{\epsilon}_{GCA}$
	21×151	Train	Test	Horizontal velocity u_x	4.6×10^{-3}
Navier-Stokes	equispaced	$r_t = 0.1$	$r_t = 0.9$	Vertical velocity u_y	2×10^{-2}
	$N_S = 3171$	$N_{tr} = 318$	$N_{te} = 2853$	Pressure p	7.7×10^{-3}

We study the steady-state Navier-Stokes system on a parametrized sudden-expansion channel geometry $\Omega(\mu_1)$, depicted in Fig. 15, discretized with $N_h = 2719$ nodes.

The domain is composed by a nozzle, with parametrized width, on which we impose an inlet velocity through a non-homogeneous Dirichlet boundary condition, and a fixed expansion chamber with a Neumann condition at the outlet.

The continuous formulation of the Navier-Stokes problem reads as: given $\boldsymbol{\mu} \in \mathbb{P}$, find the velocity and pressure fields $\mathbf{u}(\boldsymbol{\mu}) = (u_x(\boldsymbol{\mu}), u_y(\boldsymbol{\mu}))$ and $p(\boldsymbol{\mu})$, such that

$$\begin{cases} -\mu_2 \Delta \mathbf{u}(\boldsymbol{\mu}) + (\mathbf{u}(\boldsymbol{\mu}) \cdot \nabla)\, \mathbf{u}(\boldsymbol{\mu}) + \nabla p(\boldsymbol{\mu}) = 0 & \text{in } \Omega(\mu_1), \\ \nabla \cdot \mathbf{u}(\boldsymbol{\mu}) = 0 & \text{in } \Omega(\mu_1), \\ \mathbf{u}(\boldsymbol{\mu}) = \mathbf{u}_{\mathrm{in}} & \text{on } \Gamma_{\mathrm{in}}, \\ \mathbf{u}(\boldsymbol{\mu}) = \mathbf{0} & \text{on } \Gamma_{\mathrm{w}}(\mu_1), \\ \mu_2 \frac{\partial \mathbf{u}}{\partial \mathbf{n}}(\boldsymbol{\mu}) - p(\boldsymbol{\mu})\,\mathbf{n} = 0 & \text{on } \Gamma_{\mathrm{out}}, \end{cases}$$

where $\mathbf{n}$ is the normal unit vector, μ_1 is the half-width of the inlet channel, μ_2 the kinematic viscosity of the fluid, and $\mathbf{u}_{\mathrm{in}}(\boldsymbol{x}) = \left[20(5-y)(y-2.5),\ 0\right]$ the Poiseuille inflow profile.

The Navier-Stokes benchmark is parametrized by means of the physical and geometric multi-parameter $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{P} = [0.5, 2]^2$. The parametric investigation of this model is physically interesting, and the bifurcation point depends on the physical and geometrical flow's configuration. This means that being able to predict the evolution of the bifurcation point in the parameter space allows to detect the changes in the stability properties of the system.

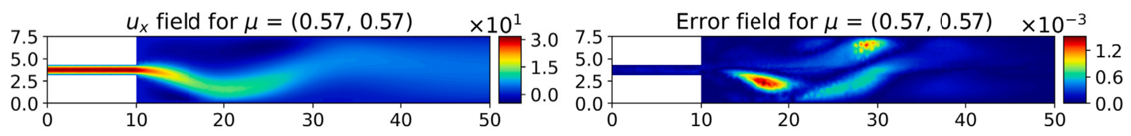
Given the nature of the model, with velocity and the pressure fields as unknowns, we show the adaptability of the architecture to vector problems. Thus, we consider a monolithic approach, where the three fields are concatenated as features for each node, rather than a partitioned one, where we recover independently each field.[6]

We report in Table 1 the parametric setting for this benchmark. We highlight that the reason for such a high number of snapshots, $N_S = 3171$, is that the Finite Element approximation of the bifurcating phenomenon requires a continuation technique to follow the symmetry-breaking branch [56]. Despite the increased cardinality of the sampling w.r.t. the former test cases, we fixed the training rate to only consider $r_t = 10\%$ of the original dataset.

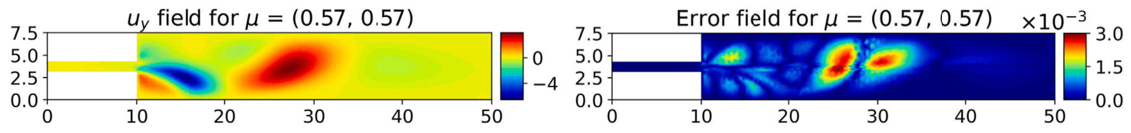
In Fig. 16 we plot the three components of the Navier-Stokes solution for $\boldsymbol{\mu} = (0.57, 0.57) \in \Xi_{\mathrm{te}}$, and the GCA-ROM relative errors w.r.t. the high-fidelity snapshots. The errors are mainly localized in the regions where the flow suddenly changes, and among the three components, the vertical velocity is the one reconstructed least accurately. The components recover the wall-hugging behavior with the flow attaching to the bottom boundary, upon which we built the dataset.

Fig. 17 depicts the relative errors $\epsilon_{GCA}(\boldsymbol{\mu})$ for the velocity and pressure components over the dataset Ξ. We observe that, even in the vector test case featuring a bifurcating behavior, the GCA-ROM architecture accurately reconstructs all the three components of the Navier-Stokes equations with small mean relative errors, as reported in Table 1.

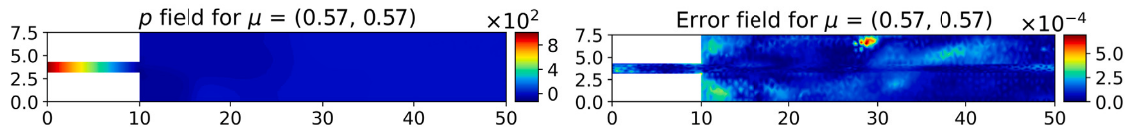
[6] Notice that this is possible given the data-driven nature of the architecture, i.e. we are not exploiting the physics coming from the PDE as in projection-based models.



(a) Horizontal component of the velocity field - u_x

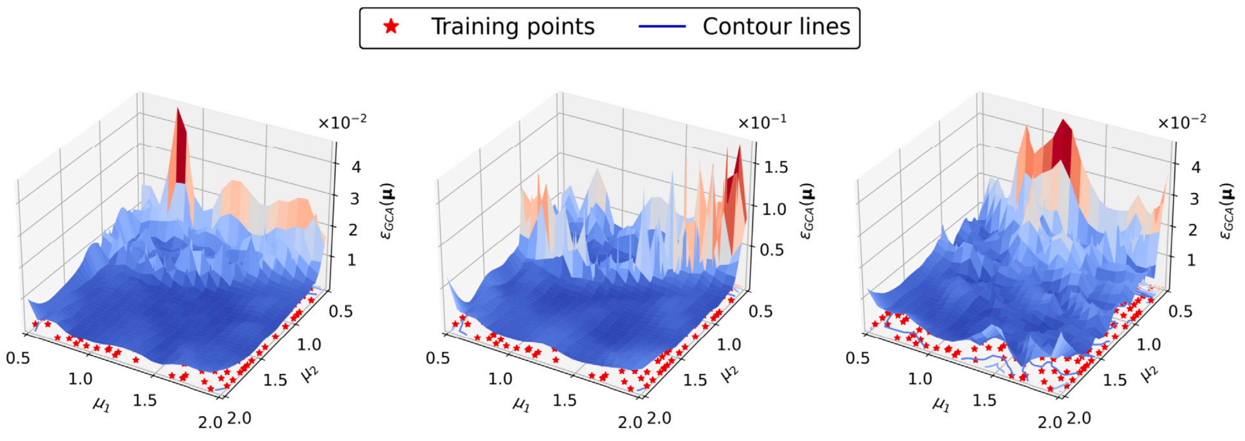


(b) Vertical component of the velocity field - u_y



(c) Pressure component - p

Fig. 16. Solution and error fields for the components of the Navier-Stokes system for $\boldsymbol{\mu} = (0.57, 0.57)$.



(a) Horizontal velocity - u_x (b) Vertical velocity - u_y (c) Pressure - p

Fig. 17. GCA-ROM relative errors for the components of the Navier-Stokes test case on the dataset Ξ, with red markers corresponding to the parameters used in the training set Ξ_{tr}.

On one side, we notice that this could still be considered a low-data regime, especially in the bifurcating context, while on the other this shows that the architecture can benefit when increasing the cardinality of Ξ_{tr} without overfitting. As observed previously, the vertical velocity field is the most difficult to approximate, due to the fact that it is the component responsible for the symmetry breaking phenomenon, and thus the bifurcation. Higher generalization errors occur again for smaller values of the viscosity. For the vertical velocity, this is maximized for bigger inlet's width, since in that region the geometry still does not admit the wall-hugging profile, while the (bifurcation-agnostic) data-driven approach is influenced by the neighboring training snapshots.

4.4.1. Clustering and classification task for bifurcating problems

Here, we show a simple yet important exploitation of the low-dimensional manifold encoded in the GCA-ROM architecture's bottleneck. In the bifurcating scenario, from structural to fluid mechanics, deciphering whether a combination of parameters belongs to the critical regime is of importance. Knowing the evolution of the low-dimensional manifold for such problems can help the detection of the bifurcating behavior without performing costly many simulations. Having built a computationally cheap mapping from the parameter to the latent space, the idea is to perform a clustering of the bottleneck's dataset to assign labels for the different states, and a classification task to predict the state's properties (bifurcation or uniqueness).

This approach has been explored within linear and non-intrusive ROMs [11]. Here, we exploit the connection between the reduced coefficients in POD-based techniques and the compressed latent information obtained through the nonlinear encoding. The

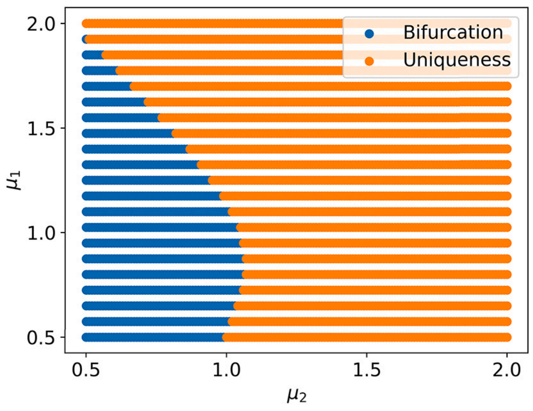


Fig. 18. Partitioning of the parameter space in bifurcating and non-bifurcating regimes, obtained with the clustering of the latent manifold based on the GCA-ROM reconstruction of u_y.

availability of real-time evaluations of the bottleneck allows to finely sample the parameter space, and obtain an accurate description of its evolution.

We choose the spectral clustering [83], which exploits eigenvalues' information of the dataset's similarity matrix. Performing the clustering in the similarity space instead of the Euclidean one, this technique is advantageous for problems with irregular and/or unbalanced clusters. In order to distinguish between bifurcating and the uniqueness regime, we only need two clusters, to identify the bifurcation points' curve in $\mathbb{P}$. Given the non-trivial behavior of the bifurcation curve, due to the relation between the viscosity and the inlet's width, we apply k-NN as classification algorithm for its efficiency in learning nonlinear boundaries. Based on these clustering results, we trained the classifier within a range of 20% to 90% of the labels, and we noticed that the accuracy was always greater than 99.5%.

In Fig. 18, we show the clustered parameter space, clearly identifying the two regimes and detecting the evolution of the bifurcation points. We notice that such a curve was already present in the error analysis in Fig. 17, identified by the maximum locations of the GCA-ROM error for the vertical component of the velocity. This means that, breaking the symmetry, the information from u_y can produce more accurate clustering. Finally, we note that the plot in Fig. 18 is in complete agreement with the results obtained with the Reduced Manifold based Bifurcation diagram (RMB) technique in [11].

4.5. *Comparison with linear and nonlinear approaches*

In this section, we compare the performance of GCA-ROM with standard linear techniques based on POD, and the original nonlinear DL-ROM architecture. Rather than focusing on the best case scenario, we detail the key features and main drawbacks of these approaches, considering the three scalar benchmarks previously discussed. As metrics, we do not only consider the mean relative error over the testing dataset Ξ_{te}, but also the offline and online timings, as well as the dimension of the optimization problem.

In Table 2, we report the accuracy over Ξ_{te} of the different methodologies when trained on the same set of snapshots Ξ_{tr} with $r_{\mathrm{t}} = 30\%$. To have a fair comparison, i.e. based on the same amount of information, we considered the GCA-ROM architecture without down-sampling, and we fixed the reduced/latent dimension to $n = 15$ for all benchmarks and methodologies.

In particular, we consider the projection error in the POD space, denoted with (POD), the standard POD-Galerkin approach (POD-G), DL-ROM and GCA-ROM. As expected, when considering a large enough latent dimension, thanks to the exponential decay of the reduced error, projecting in the POD space gives the best results in terms of approximation accuracy. POD-G still produces better performance as compared to ML-based techniques (at least for large enough values of n) when dealing with easily reducible problems, such as Poisson or Graetz.

On the contrary, nonlinear techniques show greater capabilities when dealing with advection-dominated phenomena, which are characterized by a slow Kolmogorov n-width decay. In all cases, GCA-ROM provided better results than its CNN-based counterpart, even beating the POD-G approach for the Advection test case. This confirms that augmenting the learning procedure with geometrical biases helps the optimization step.

Table 2
Mean relative errors of the scalar benchmarks over Ξ_{te} for POD with projection, POD-G, DL-ROM and GCA-ROM techniques, with reduced/latent dimension $n = 15$.

Application	POD	POD-G	DL-ROM	GCA-ROM
Poisson	9.9×10^{-5}	1.0×10^{-4}	1.5×10^{-2}	7.8×10^{-3}
Advection	3.1×10^{-2}	4.0×10^{-2}	5.0×10^{-2}	2.4×10^{-2}
Graetz	2.3×10^{-4}	2.4×10^{-4}	1.4×10^{-2}	6.8×10^{-3}

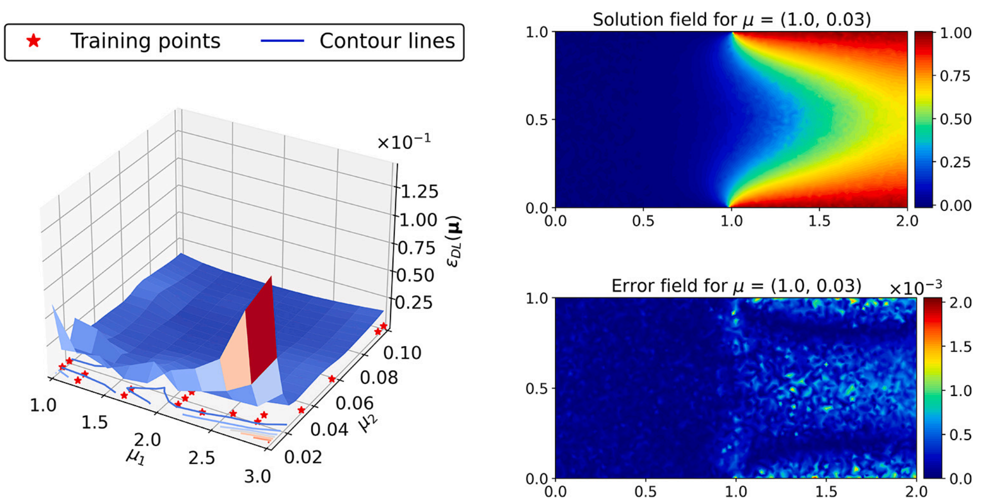


Fig. 19. DL-ROM relative error for the Graetz problem on the dataset Ξ, and solution and error fields for $\boldsymbol{\mu} = (1, 0.03)$, left and right respectively.

Table 3
Comparison between the number of trainable parameters, training, and testing times for DL-ROM and GCA-ROM with different filter sizes.

Method	Device	Filters	Parameters	Training time (s)	Testing time (s)
DL-ROM	CPU	3x3	8 476 109	66 518	9.49
		5x5	6 592 461	62 060	9.74
	GPU	3x3	8 476 109	1172	8.93
		5x5	6 592 461	1323	9.29
GCA-ROM	CPU	3	2 088 682	3967	13.94
		5	2 088 694	6944	14.86
	GPU	3	2 088 682	553	15.43
		5	2 088 694	714	14.92

Other non-intrusive POD-based approaches, such as POD-NN or PODI, have been investigated, but the low-data regime we are interested in is a well-known issue for these methodologies. Indeed, both non-intrusive regression and interpolation approaches usually require a large training dataset, without which the learning capabilities are compromised, and the model is unable to generalize.

As an example, we show the DL-ROM results for the geometrically parametrized Graetz problem. Relying on an affine map transformation between the original and reference domains, this corresponds to the easiest setting in which to investigate the performance for unstructured grids in advanced applications. In Fig. 19 we plot the relative error of the DL-ROM approach $\varepsilon_{DL}(\boldsymbol{\mu})$, and the solution and error fields of the Graetz model obtained for $\boldsymbol{\mu} = (1, 0.03) \in \Xi_{\text{te}}$. While the original approach is still able to learn the main features of the model, with possible implications still to be understood [84,85], and a mean $\bar{\varepsilon}_{DL} = 1.6 \times 10^{-2}$, we notice that it also produces an unsatisfactory maximum one around 1.4×10^{-1}. This is localized in the area corresponding to the maximum stretching for the domain, where the geometric consistency is even more crucial.

Furthermore, given the non-locality of the convolutional operator in DL-ROM, the reconstruction of the solution field is in general not smooth, as the one depicted in Fig. 19. The error shows a speckled pattern affecting the solution in its entire domain, and producing an oscillating reconstruction of the thermal field. Comparing the results with the ones in Fig. 13, the same effect is much less evident, producing a smooth field even considering a highly deformed geometrical configuration and a lower viscosity value.

As regard the efficiency, we now discuss both the offline and online complexity of the two methodologies. While the GCA-ROM approach takes as input the high-fidelity fields already defined on their unstructured geometry, DL-ROM approaches require reshaping the data as a matrix, applying a padding strategy to reach the desired dimensions and apply convolution. This artificial addition of zeroes could potentially compromise the learning procedure, as well as increasing its computational complexity.

In Table 3 we compare GCA-ROM and DL-ROM in terms of training and testing times, and number of parameters involved in the optimization process. It is worth noting that message passing is a computationally demanding operation for GNNs, but exploiting local convolutions in such context, the sharing of weights reduces significantly the amount of learnable parameters in the network. Depending on the dimension of the filters, we notice a reduction of order three to four. Indeed, the amount of weights exploited by DL-ROM heavily depends on the filter sizes, while for GCA-ROM it is almost the same, i.e. the cost of the weighting functions in the Gaussian kernels of MoNet is dominated by the one of the feed-forward layer. For this reason, a more in depth investigation of the up-sampling and down-sampling procedures, could drastically improve the efficiency of the presented methodology.

All simulations have been performed on a workstation equipped with an Nvidia Quadro GP100 GPU. We report both CPU and GPU times, showing that GCA-ROM is computationally affordable even when only modest hardware is available.[7] Testing times are including the computation of the reconstructed solutions for all the parameter values in Ξ.

5. Conclusions

This work showcases the performance of deep learning based on graph neural networks to learn reduced representations of PDEs' solutions. The use of geometric information through networks helps to process unstructured information properly. Introducing a consistent way of treating complex and potentially parametrized geometries, the proposed GCA-ROM architectures improve the generalization property of deep learning approaches, thus paving the way for the analysis of many developments based on graph neural networks.

We tested our novel methodology on a diverse set of parametric problems, exhibiting different characteristics for challenging linear and nonlinear reduced order approaches. In particular, we considered both complex physical behaviors, as advection-dominated problems and bifurcating systems, and unstructured non-rectangular domains, characterized by parameter dependent geometries. In all scenarios the methodology showed great accuracy, even in the low-data regime, exploiting as little as the 30 % of the original dataset.

GCA-ROM was compared with the original DL-ROM approach, highlighting the advantages of adopting graph-based operations in deep learning task related to PDE's solutions. We achieved better accuracy while reducing the computational cost and number of trainable parameters. As a result, the method learns the general pattern of the graphs, thus preventing the autoencoder from overfitting. Given the diffusive nature of the message passing algorithm, based on localized convolutions, GCA-ROM seems to learn smoother fields as compared to standard CNN-based approaches.

The application of convolutional layers, particularly of those defined on graphs, shows robust performance while learning across different mesh topologies. In addition, the chosen graph convolution operation, MoNet, incorporates the information regarding the node's position, and the edge attributes as an attention mechanism. These two key attributes enable the method to adapt to any topological configuration efficiently. However, the current algorithm relies on a fixed size of the layers for the fully connected multilayer perceptron, connecting the encoder/decoder structures to the latent space, thus requiring fixed input/output shapes. A way to solve this issue could be to exploit the pooling operation to reduce the mesh to a fixed number of nodes, and no longer to a percentage r_p of the original dimension.

We considered a plain GCA-ROM architecture, and its pooling-based version comprising also down- and up-sampling operations to coarsen and refine the mesh during learning. We stress that pooling procedures are a valuable tool to extract features from datasets, and effectively reduce the dimensionality without adding further complexity to the network. However, these operations are still in the early stage of development, imposing limitations especially in the decoder structure, where costly interpolations have to be computed. Despite this, such approaches could drastically decrease the size of the optimization problem, and their analysis could create bridges with several methodologies, such as multigrid methods [86], physics-informed approaches [87,88] and time-integration schemes. Another interesting direction is towards three-dimensional problems, with promising explorations for cardiovascular and musculoskeletal applications [89–91]. Future work will include high-dimensional parameter space, the enforcement of boundary conditions and the combination of GNNs with different discretization techniques, exploiting even further their versatility.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Federico Pichi reports financial support was provided by Rectors' Conference of Italian Universities. Beatriz Moya reports financial support was provided by Ibercaja Banking Foundation. Beatriz Moya reports financial support was provided by Spain Ministry of Science and Innovation, grant number PID2020-113463RB-C31.

Code availability

https://github.com/fpichi/gca-rom.

Data availability

I have shared the link to the GitHub page in the abstract, where codes and datasets are collected for reproducibility purposes.

Acknowledgements

Federico Pichi acknowledges the support by "GO for IT" program within a CRUI fund for the project "Reduced order method for nonlinear PDEs enhanced by machine learning". Beatriz Moya acknowledges the support of the University of Zaragoza, the bank

[7] We remark that computational GPU times in the GCA-ROM setting can be further optimized by exploiting new releases of PyTorch Geometric, enabling even faster computations.

institution Ibercaja, and the institution CAI (project reference IT1/21). She also thanks the Ministry of Science and Innovation of the Government of Spain (project reference AEI/10.13039/501100011033 with grant number PID2020-113463RB-C31).

Appendix A. Architectures

A.1. Architectures and hyperparameters' analysis

This section summarizes the GCA-ROM structures used for the numerical results, the hyperparameter investigation to understand the properties of the architecture w.r.t. each module, and the corresponding best network configurations for the reproducibility of the results.

We briefly recall the notations used to define the methodology. The dataset is formed by N_S snapshots, i.e. the high-fidelity solutions with $d \geq 1$ components (scalar or vector problem) defined over the N_h nodes of the mesh. The training rate r_t determines the amount of data available for the learning task, while the pooling rate r_p defines the amount of sensors exploited for the down- and up-sampling procedures. As regard the autoencoder architecture, we have hcp a-priori convolutional layers, and hcd a-posteriori convolutional layers when the pooling module is considered. The connection with the bottleneck of size n is obtained through a 2-layers fully-connected neural network (FC) of size ffn. The parameter map, needed for the online evaluation, is formed by an MLP with n_l neurons in each of the 5-layers. The parameter λ is the weight in the loss term balancing the autoencoder and the MLP. The dimensionality of the parameter space $\mathbb{P}$ is equal to $P = 2$ for all benchmarks. The learning rate and the weight decay have been fixed throughout all the simulations to the values $l_\text{r} = 10^{-3}$ and $w_\text{d} = 10^{-5}$. In Table A.4 we schematize the GCA-ROM architecture described in Figs. 4 and 5, and used for the scalar and vector benchmarks. The shaded area corresponds to the pooling module which can be enabled. Otherwise, we consider $r_\text{p} = 100\%$.

We show in Figs. A.20 and A.21 the hyperparameter investigations obtained with the plain GCA-ROM approach for the Poisson problem, and the GCA-ROM with pooling for the Advection benchmark, respectively. The former illustrates the box-plot of the mean relative errors $\varepsilon_{GCA}(\boldsymbol{\mu})$ for all possible combinations of the following set of hyperparameters: $r_\text{t} \in [10, 30, 50]$, $r_\text{p} \in [30, 50, 70]$, ffn $\in [100, 200, 300]$, $n_l \in [50, 100]$, $n \in [15, 25]$, $\lambda \in [0.1, 1, 10]$, and hc $\in [1, 2, 3]$

In Fig. A.21 we plot the same quantity, fixing the number of nodes in the latent map $n_l = 50$, the dimension of the bottleneck $n = 25$, and the weight of the loss $\lambda = 1$, while varying: $r_\text{p} \in [30, 50, 70]$, $r_t \in [10, 30, 50]$, ffn $\in [100, 200]$, and hcp, hcd $\in [1, 2, 3]$.

Finally, we report in Table A.5 the best configuration, for the plain and pooling versions of GCA-ROM, resulting from the aforementioned hyperparameter analysis, and which we have used throughout the manuscript.

Table A.4
Autoencoder architecture for GCA-ROM approach, with optional pooling module in gray.

Module	Layers	Input size	Output size	Activation	Kernel
Encoder	Conv_hcp	$n_b \times N_h \times d$	$n_b \times N_h \times d$	ELU	Q
	Pool	$n_b \times N_h \times d$	$n_b \times r_\text{p} N_h \times d$		
	Conv_hcd	$n_b \times d r_\text{p} N_h$	$n_b \times d r_\text{p} N_h$	ELU	Q
	FC	$n_b \times d r_\text{p} N_h$	$n_b \times$ fnn	ELU	
		$n_b \times$ fnn	$n_b \times n$	Identity	
Decoder	FC	$n_b \times n$	$n_b \times$ fnn	Identity	
		$n_b \times$ fnn	$n_b \times d r_\text{p} N_h$	ELU	
	Conv_hcd	$n_b \times d r_\text{p} N_h$	$n_b \times d r_\text{p} N_h$	ELU	Q
	Unpool	$n_b \times r_\text{p} N_h \times d$	$n_b \times N_h \times d$		
	Conv_hcp	$n_b \times N_h \times d$	$n_b \times N_h \times d$	ELU	Q
Parameter map	FC	$n_b \times P$	$n_b \times n_l$	tanh	
		⋮	⋮	⋮	
		$n_b \times n_l$	$n_b \times n$	Identity	

Table A.5
Hyperparameter configuration of the best network for each benchmark.

Application	pooling	r_p	r_t	λ	btt	hcp	hcd	nd	ffn
Poisson	✗	✗	30	10	15	3	✗	50	100
	✓	70	30	1	25	3	3	50	200
Advection	✗	✗	30	10	15	2	✗	100	200
	✓	70	30	1	25	3	3	50	200
Graetz	✗	✗	30	10	25	2	✗	50	200
	✓	70	30	1	25	2	1	50	100
Navier-Stokes	✗	✗	10	1	25	3	✗	100	200

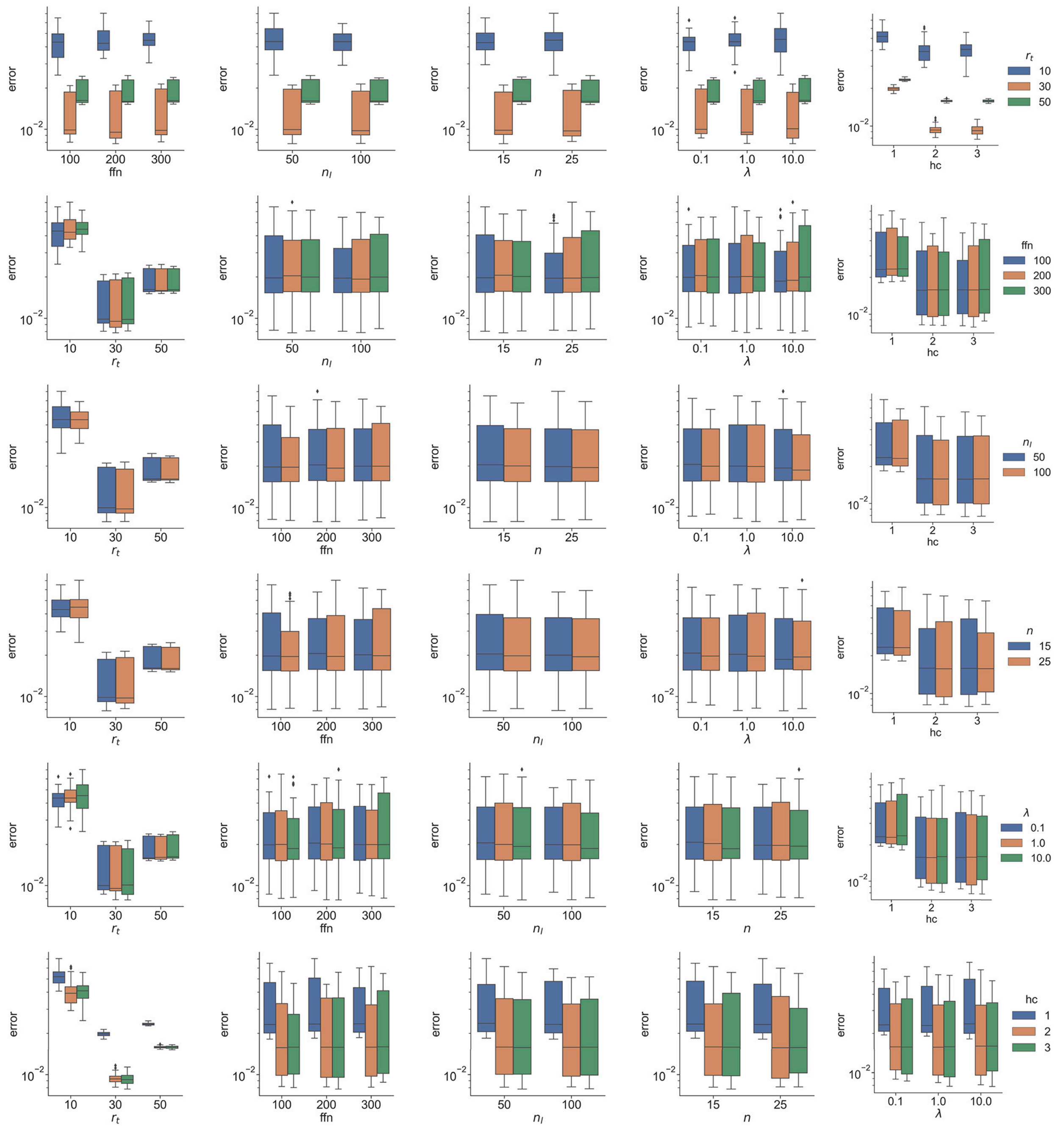


Fig. A.20. Box-plot of the mean relative error $\varepsilon_{GCA}(\boldsymbol{\mu})$ for the Poisson test case.

References

[1] P. Benner, S. Grivet Talocia, A. Quarteroni, G. Rozza, W. Schilders, L.M. Silveira, Model Order Reduction, Vol. 1-3, De Gruyter, 2020.
[2] P. Benner, A. Cohen, M. Ohlberger, K. Willcox, Model Reduction and Approximation: Theory and Algorithms, Computational Science and Engineering Series, SIAM, Society for Industrial and Applied Mathematics, 2017, https://books.google.it/books?id=hbEsDwAAQBAJ.
[3] J.S. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, 1st Edition, SpringerBriefs in Mathematics, Springer International Publishing AG, Cham, 2015.
[4] A. Quarteroni, A. Manzoni, F. Negri, Reduced Basis Methods for Partial Differential Equations: An Introduction, 1st Edition, La Matematica per Il, vol. 3+2, 92, Springer International Publishing, Cham, 2016.
[5] K. Lee, K.T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, J. Comput. Phys. 404 (2020) 108973.
[6] S. Fresca, L. Dede, A. Manzoni, A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs, J. Sci. Comput. 87 (2) (2021) 1–36.

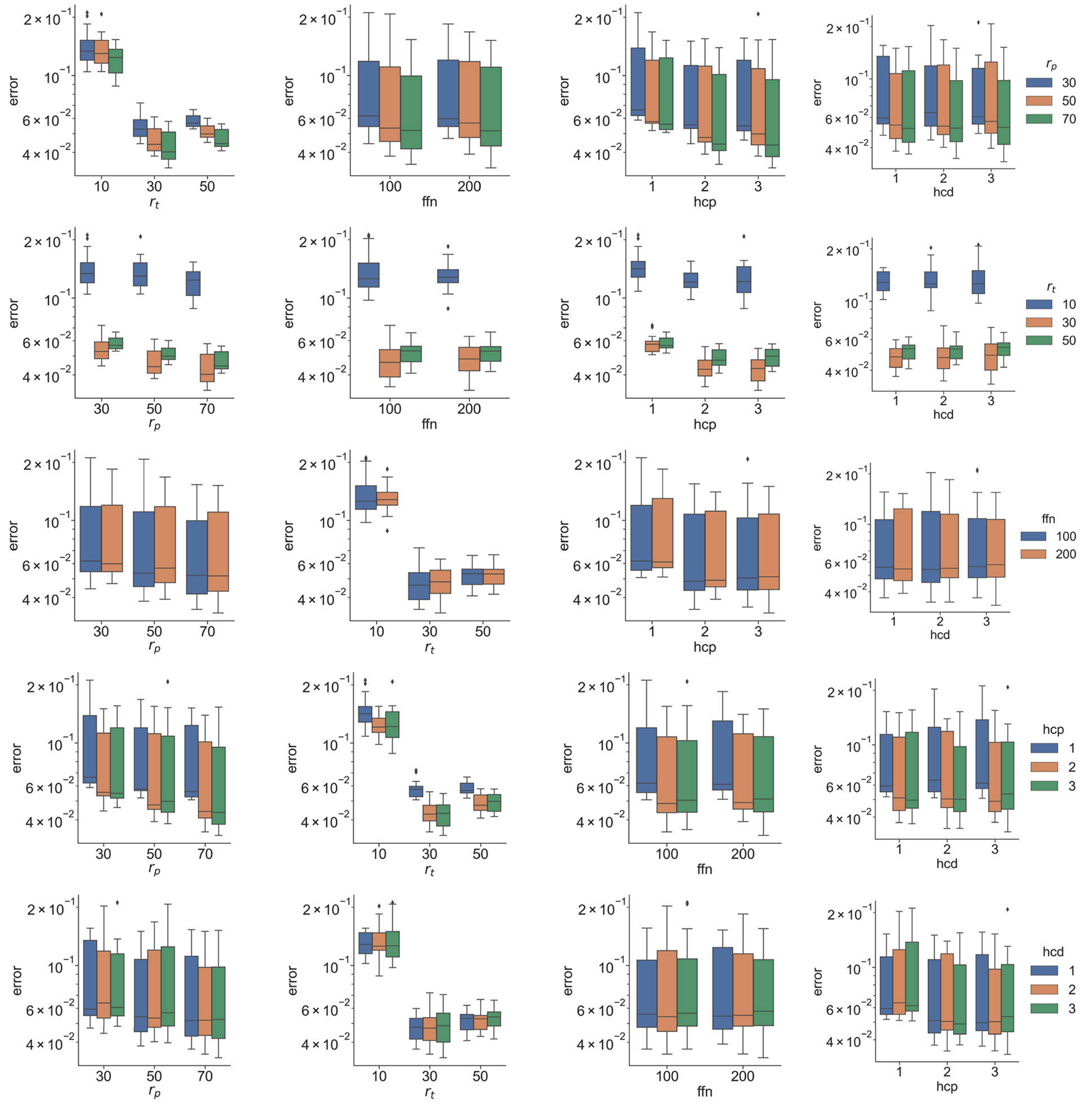


Fig. A.21. Box-plot of the mean relative error $\varepsilon_{GCA}(\boldsymbol{\mu})$ for the Advection test case with pooling strategy.

[7] R. Vinuesa, S.L. Brunton, Enhancing computational fluid dynamics with machine learning, Nat. Comput. Sci. 2 (6) (2022) 358–366.
[8] M. Milano, P. Koumoutsakos, Neural Network Modeling for Near Wall Turbulent Flow, J. Comput. Phys. 182 (1) (2002) 1–26, https://doi.org/10.1006/jcph.2002.7146.
[9] Q. Hernandez, A. Badias, D. Gonzalez, F. Chinesta, E. Cueto, Deep learning of thermodynamics-aware reduced-order models from data, Comput. Methods Appl. Mech. Eng. 379 (2021) 113763.
[10] J.S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, J. Comput. Phys. 363 (2018) 55–78.
[11] F. Pichi, F. Ballarin, G. Rozza, J.S. Hesthaven, An artificial neural network approach to bifurcating phenomena in computational fluid dynamics, Comput. Fluids 254 (2023) 105813, https://doi.org/10.1016/j.compfluid.2023.105813.
[12] 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) 037106.
[13] J.-Z. Peng, S. Chen, N. Aubry, Z. Chen, W.-T. Wu, Unsteady reduced-order model of flow over cylinders based on convolutional and deconvolutional neural network structure, Phys. Fluids 32 (12) (2020) 123609.
[14] 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) 077107.
[15] S. Pawar, S.E. Ahmed, O. San, A. Rasheed, Data-driven recovery of hidden physics in reduced order modeling of fluid flows, Phys. Fluids 32 (3) (2020) 036602.
[16] M. Morimoto, K. Fukami, K. Zhang, A.G. Nair, K. Fukagata, Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization, Theor. Comput. Fluid Dyn. 35 (5) (2021) 633–658.

[17] A. Sharma, E. Vans, D. Shigemizu, K.A. Boroevich, T. Tsunoda, Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture, Sci. Rep. 9 (1) (2019) 1–7.
[18] N.R. Franco, A. Manzoni, P. Zunino, Learning Operators with Mesh-Informed Neural Networks, arXiv preprint, arXiv:2203.11648, 2022.
[19] D. Coscia, L. Meneghetti, N. Demo, G. Stabile, G. Rozza, A continuous convolutional trainable filter for modelling unstructured data, arXiv preprint, arXiv:2210.13416, 2022.
[20] K. Doherty, C. Simpson, S. Becker, A. Doostan, QuadConv: Quadrature-Based Convolutions with Applications to Non-Uniform PDE Data Compression, arXiv preprint, arXiv:2211.05151, 2023.
[21] M.M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep learning: grids, groups, graphs, geodesics, and gauges, arXiv preprint, arXiv:2104.13478, 2021.
[22] 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, arXiv preprint, arXiv:1806.01261, 2018.
[23] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint, arXiv:1609.02907, 2016.
[24] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M.M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model cnns, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5115–5124.
[25] FEniCS, https://www.fenicsproject.org.
[26] RBniCS, https://www.rbnicsproject.org.
[27] M. Fey, J.E. Lenssen, Fast graph representation learning with PyTorch Geometric, in: ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.
[28] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81, https://doi.org/10.1016/j.aiopen.2021.01.001.
[29] W.L. Hamilton, Graph representation learning, Synth. Lect. Artif. Intell. Mach. Learn. 14 (3) (2020) 1–159.
[30] 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.
[31] M. Xu, S. Song, X. Sun, W. Zhang, A convolutional strategy on unstructured mesh for the adjoint vector modeling, Phys. Fluids 33 (3) (2021) 036115.
[32] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.
[33] Q. Hernandez, A. Badias, F. Chinesta, E. Cueto, Thermodynamics-informed graph neural networks, IEEE Trans. Artif. Intell. (2022).
[34] M. Fortunato, T. Pfaff, P. Wirnsberger, A. Pritzel, P. Battaglia, MultiScale MeshGraphNets, arXiv preprint, arXiv:2210.00612, 2022.
[35] R.J. Gladstone, H. Rahmani, V. Suryakumar, H. Meidani, M. D'Elia, A. Zareei, GNN-based physics solver for time-independent PDEs, arXiv preprint, arXiv:2303.15681, 2023.
[36] X. Han, H. Gao, T. Pffaf, J.-X. Wang, L.-P. Liu, Predicting physics in mesh-reduced space with temporal attention, arXiv preprint, arXiv:2201.09113, 2022.
[37] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Trans. Graph. 38 (5) (2019) 1–12.
[38] H. Gao, S. Ji, Graph U-nets, in: international conference on machine learning, PMLR, 2019, pp. 2083–2092.
[39] A. Ranjan, T. Bolkart, S. Sanyal, M.J. Black, Generating 3d faces using convolutional mesh autoencoders, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 704–720.
[40] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. 29 (2016).
[41] Y. Zhou, C. Wu, Z. Li, C. Cao, Y. Ye, J. Saragih, H. Li, Y. Sheikh, Fully convolutional mesh autoencoder using efficient spatially varying kernels, Adv. Neural Inf. Process. Syst. 33 (2020) 9251–9262.
[42] 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) 027104.
[43] A. Gruber, M. Gunzburger, L. Ju, Z. Wang, A comparison of neural network architectures for data-driven reduced-order modeling, Comput. Methods Appl. Mech. Eng. 393 (2022) 114764.
[44] C. Cangea, P. Veličković, N. Jovanović, T. Kipf, P. Liò, Towards sparse hierarchical graph classifiers, arXiv preprint, arXiv:1811.01287, 2018.
[45] J. Qin, L. Liu, H. Shen, D. Hu, Uniform pooling for graph networks, Appl. Sci. 10 (18) (2020) 6287.
[46] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, Adv. Neural Inf. Process. Syst. 30 (2017).
[47] P.G. Ciarlet, Linear and Nonlinear Functional Analysis with Applications, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2013, https://books.google.it/books?id=AUlWAQAAQBAJ.
[48] S. Volkwein, Model reduction using proper orthogonal decomposition, in: Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz, 2011, p. 1025.
[49] M. Barrault, N.C. Nguyen, Y. Maday, A.T. Patera, An "empirical interpolation" method: Application to efficient reduced-basis discretization of partial differential equations, C. R. Acad. Sci. Paris, Ser. I 339 (2004) 667–672.
[50] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (5) (2010) 2737–2764.
[51] T. Bui-Thanh, M. Damodaran, K. Willcox, Proper Orthogonal Decomposition Extensions for Parametric Applications in Compressible Aerodynamics, in: 21st AIAA Applied Aerodynamics Conference, Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, 2003.
[52] N. Demo, M. Tezzele, G. Rozza, A non-intrusive approach for the reconstruction of POD modal coefficients through active subspaces, C. R., Méc. 347 (11) (2019) 873–881, https://doi.org/10.1016/j.crme.2019.11.012.
[53] J.L. Barnett, C. Farhat, Y. Maday, Neural-Network-Augmented Projection-Based Model Order Reduction for Mitigating the Kolmogorov Barrier to Reducibility of CFD Models, arXiv preprint, arXiv:2212.08939, 2022.
[54] W. Chen, Q. Wang, J.S. Hesthaven, C. Zhang, Physics-informed machine learning for reduced-order modeling of nonlinear problems, J. Comput. Phys. 446 (2021) 110666, https://doi.org/10.1016/j.jcp.2021.110666.
[55] C. Greif, K. Urban, Decay of the Kolmogorov N-width for wave problems, Appl. Math. Lett. 96 (2019) 216–222, https://doi.org/10.1016/j.aml.2019.05.013.
[56] F. Pichi, Reduced order models for parametric bifurcation problems in nonlinear PDEs, Ph.D. thesis, Scuola Internazionale Superiore di Studi Avanzati, 2020.
[57] T. Taddei, A Registration Method for Model Order Reduction: Data Compression and Geometry Reduction, SIAM J. Sci. Comput. 42 (2) (2020) A997–A1027, https://doi.org/10.1137/19M1271270.
[58] P. Díez, A. Muixí, S. Zlotnik, A. García-González, Nonlinear dimensionality reduction for parametric problems: A kernel proper orthogonal decomposition, Int. J. Numer. Methods Eng. 122 (24) (2021) 7306–7327, https://doi.org/10.1002/nme.6831.
[59] M. Ohlberger, S. Rave, Nonlinear reduced basis approximation of parameterized evolution equations via the method of freezing, C. R. Math. 351 (23) (2013) 901–906, https://doi.org/10.1016/j.crma.2013.10.028.
[60] J. Reiss, P. Schulze, J. Sesterhenn, V. Mehrmann, The Shifted Proper Orthogonal Decomposition: A Mode Decomposition for Multiple Transport Phenomena, SIAM J. Sci. Comput. 40 (3) (2018) A1322–A1344, https://doi.org/10.1137/17M1140571.
[61] D. Amsallem, M.J. Zahr, C. Farhat, Nonlinear model order reduction based on local reduced-order bases, Int. J. Numer. Methods Eng. 92 (10) (2012) 891–916, https://doi.org/10.1002/nme.4371.
[62] V. Ehrlacher, D. Lombardi, O. Mula, F.-X. Vialard, Nonlinear model reduction on metric spaces. Application to one-dimensional conservative PDEs in Wasserstein spaces, ESAIM: Math. Model. Numer. Anal. 54 (6) (2020) 2159–2197, https://doi.org/10.1051/m2an/2020013.

[63] F. Romor, G. Stabile, G. Rozza, Non-linear Manifold Reduced-Order Models with Convolutional Autoencoders and Reduced Over-Collocation Method, J. Sci. Comput. 94 (3) (2023) 74, https://doi.org/10.1007/s10915-023-02128-2.
[64] S. Fresca, A. Manzoni, POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition, Comput. Methods Appl. Mech. Eng. 388 (2022) 114181.
[65] L. Cicci, S. Fresca, A. Manzoni, Deep-HyROMnet: A Deep Learning-Based Operator Approximation for Hyper-Reduction of Nonlinear Parametrized PDEs, J. Sci. Comput. 93 (2) (2022) 57, https://doi.org/10.1007/s10915-022-02001-8.
[66] B. Moya, A. Badias, D. Gonzalez, F. Chinesta, E. Cueto, Physics perception in sloshing scenes with guaranteed thermodynamic consistency, IEEE Trans. Pattern Anal. Mach. Intell. (2022).
[67] M. Guo, J.S. Hesthaven, Reduced order modeling for nonlinear structural analysis using Gaussian process regression, Comput. Methods Appl. Mech. Eng. 341 (2018) 807–826, https://doi.org/10.1016/j.cma.2018.07.017.
[68] B. Peherstorfer, K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, Comput. Methods Appl. Mech. Eng. 306 (2016) 196–215, https://doi.org/10.1016/j.cma.2016.03.025.
[69] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Learning maps between function spaces, arXiv preprint, arXiv:2108.08481, 2021.
[70] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229, https://doi.org/10.1038/s42256-021-00302-5.
[71] N. Demo, M. Tezzele, G. Rozza, A DeepONet Multi-Fidelity Approach for Residual Learning in Reduced Order Modeling, arXiv preprint, arXiv:2302.12682, 2023.
[72] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).
[73] S. Berrone, F. Della Santa, A. Mastropietro, S. Pieraccini, F. Vaccarino, Graph-Informed Neural Networks for Regressions on Graph-Structured Data, Mathematics 10 (5) (2022) 786, https://doi.org/10.3390/math10050786.
[74] F.M. Bianchi, D. Grattarola, C. Alippi, Spectral clustering with graph neural networks for graph pooling, in: International Conference on Machine Learning, PMLR, 2020, pp. 874–883.
[75] C. Liu, Y. Zhan, C. Li, B. Du, J. Wu, W. Hu, T. Liu, D. Tao, Graph pooling for graph neural networks: Progress, challenges, and opportunities, arXiv preprint, arXiv:2204.07321, 2022.
[76] J. Xu, L. Zikatanov, Algebraic multigrid methods, Acta Numer. 26 (2017) 591–721.
[77] D. Grattarola, D. Zambon, F.M. Bianchi, C. Alippi, Understanding pooling in graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[78] S. Barwey, V. Shankar, V. Viswanathan, R. Maulik, Multiscale Graph Neural Network Autoencoders for Interpretable Scientific Machine Learning, arXiv preprint, arXiv:2302.06186, 2023.
[79] 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.
[80] F. Pichi, M. Strazzullo, F. Ballarin, G. Rozza, Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: Application to Navier–Stokes equations with model order reduction, ESAIM: Math. Model. Numer. Anal. 56 (4) (2022) 1361–1400, https://doi.org/10.1051/m2an/2022044.
[81] M. Khamlich, F. Pichi, G. Rozza, Model order reduction for bifurcating phenomena in fluid-structure interaction problems, Int. J. Numer. Methods Fluids 94 (10) (2022) 1611–1640, https://doi.org/10.1002/fld.5118.
[82] N. Tonicello, A. Lario, G. Rozza, G. Mengaldo, Non-intrusive reduced order models for the accurate prediction of bifurcating phenomena in compressible fluid dynamics, arXiv preprint, arXiv:2212.10198, 2022.
[83] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[84] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning (still) requires rethinking generalization, Commun. ACM 64 (3) (2021) 107–115, https://doi.org/10.1145/3446776.
[85] N.R. Franco, S. Fresca, A. Manzoni, P. Zunino, Approximation bounds for convolutional neural networks in operator learning, Neural Netw. 161 (2023) 129–141, https://doi.org/10.1016/j.neunet.2023.01.029.
[86] P.F. Antonietti, N. Farenga, E. Manuzzi, G. Martinelli, L. Saverio, Agglomeration of Polygonal Grids using Graph Neural Networks with applications to Multigrid solvers, arXiv preprint, arXiv:2210.17457, 2023.
[87] H. Gao, M.J. Zahr, J.-X. Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Eng. 390 (2022) 114502, https://doi.org/10.1016/j.cma.2021.114502.
[88] Y. Kim, Y. Choi, D. Widemann, T. Zohdi, A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder, J. Comput. Phys. 451 (2022) 110841.
[89] 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, arXiv preprint, arXiv:2212.05023, 2022.
[90] L. Pegolotti, M.R. Pfaller, N.L. Rubio, K. Ding, R.B. Brufau, E. Darve, A.L. Marsden, Learning Reduced-Order Models for Cardiovascular Simulations with Graph Neural Networks, arXiv preprint, arXiv:2303.07310, 2023.
[91] J. Kneifl, D. Rosin, O. Röhrle, J. Fehr, Low-dimensional Data-based Surrogate Model of a Continuum-mechanical Musculoskeletal System Based on Non-intrusive Model Order Reduction, arXiv preprint, arXiv:2302.06528, 2023.