Structural and Multidisciplinary Optimization (2024) 67:32
https://doi.org/10.1007/s00158-024-03771-5

INDUSTRIAL APPLICATION PAPER

Aerodynamic shape optimization using graph variational autoencoders and genetic algorithms

Jorge Jabón[1] · **Sergio Corbera**[1] · **Roberto Álvarez**[1] · **Rafael Barea**[1]

Received: 24 August 2023 / Revised: 4 January 2024 / Accepted: 12 February 2024 / Published online: 26 February 2024
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2024

Abstract

The use of machine learning in aerodynamic shape optimization problems has significantly increased in recent years. While existing deep learning techniques enable efficient design space exploration on data with an underlying Euclidean or grid-like structure, the direct optimization of non-parametric 3D geometries is still limited. In this article, we propose a geometric deep learning model that generates triangled-based meshed surfaces through the use of a graph variational autoencoder that learns the latent representations of a non-parametric 3D dataset. Once this framework is trained to embed all the input meshes in a properly distributed latent space, its exploration is managed by a genetic algorithm. In this regard, the NSGA-II is the agent in charge of sampling geometries that combine topology and aerodynamic features of the initial ones. Furthermore, in each iteration, it evaluates their aerodynamic performance with CFD in order to guide the optimization process and find the most effective region of the latent space. As a result, those solutions that maximize aerodynamic performance are provided through a Pareto front. The application to a case study and a real-world application is introduced aiming to validate the proposed approach.

Keywords Generative design · Geometric deep learning · Autoencoders · Genetic algorithms · CFD

1 Introduction

Aerodynamic shape optimization (ASO) tools are used by aerodynamicists to explore the design of bodies in which lift and drag are important features (Li et al. 2022). The aerodynamic performance is usually evaluated using computational fluid dynamics (CFD) and the optimization can be done using several methods, including gradient-based and gradient-free algorithms (Foster and Dulikravich 1997). In particular, ASO considerably reduces the development cycle time and improves the design of wings, tails, engine nacelles, turbine blades, and other components. The current workflow of ASO is described in Fig. 1 (Yan et al. 2019). It is a combination of aerodynamic calculations and optimization theories, converting the optimization problem into a highly complex and computationally expensive problem with numerical simulations (Secanell et al. 2006).

On the other hand, machine learning has emerged as a set of algorithms that learn from experience and available results, hence allowing it to solve physical problems (Bishop 2006) without solving the costly governing equations or by reducing its complexity. Specifically, deep learning (DL) is differentiated from classical approaches by the set of powerful models it focuses on (Zhang et al. 2021). These models consist of many successive data transformations chained together from top to bottom, thus they have the potential to speed up ASO. Since a large design space with hundreds of design variables is usually used (Lyu et al. 2015), which plays an important role in the convergence and computational cost of the optimization due to the course of dimensionality (Bellman 1957), recent works try to balance exploration and exploitation while sampling the design space. Thus, dimensionality reduction methods of the original design space have been proposed to derive more compact shape representations.

Most research focuses on two-dimensional (2D) representations through the use of successful neural networks that work on data with an underlying Euclidean or grid-like

Responsible editor: Gang Li

✉ Jorge Jabón
j.jabon@hotmail.com; jjabond@alumnos.nebrija.es; jjabon@nebrija.es

[1] Escuela Politécnica Superior y de Arquitectura, Universidad Nebrija, Madrid, Spain

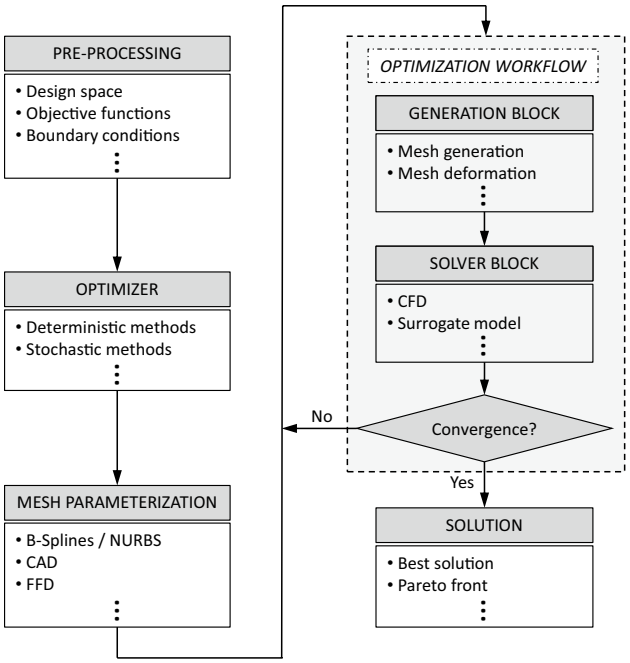

Fig. 1 ASO workflow (Yan et al. 2019). Firstly, a pre-processing step is used to define the problem to solve and the optimizer to use. Then, the geometry is parameterized to define the design variables the optimizer will use to control the shape variation. Finally, an iterative process is launched in which each candidate is generated and evaluated until convergence is reached

structure. Nevertheless, when dealing with three-dimensional (3D) representations, the application of DL for ASO is limited to dimensionality reduction with free-form deformation methods or with 2D sketches that are subsequently stacked. In this paper, we address the problem of directly optimizing 3D geometries as a whole by the application of geometric deep learning (GDL), an umbrella term for emerging techniques attempting to generalize deep neural models to non-Euclidean domains such as graphs or manifolds (Bronstein et al. 2017). In summary, the framework presented introduces a representation that models geometric variations on 3D non-parametric meshes and an optimizer that moves along the latent space to generate new geometries with interpolated features of the dataset. Our main contributions are as follows: (1) from now on, there is no need to work with totally parametric computer-aided design (CAD) files and software to produce geometric variability nor to rely on external parameterizing techniques like free-form deformation (FFD); (2) the way in which the features of the geometries are represented assures compatibility with common polygonal mesh files such as *.stl, *.ply or *.obj; (3) the well-known non-dominated sorting genetic algorithm (NSGA-II) is coupled with a graph variational autoencoder (GVAE) to handle the shape optimization given its excellent capability to solve multi-objective optimization problems (MOOP), generating a Pareto front for decision-making with the most promising geometries in terms of aerodynamic performance; (4) the use of commercial software for CFD analysis guarantees the easy integration into the optimization workflow followed by aerospace industry for optimizing real-world parts; (5) the application of the proposed framework not only allows the optimization of aerodynamic parts, but also the same procedure can be followed in structural problems.

This article is structured as follows: Sect. 2 overviews current works in aerodynamic shape optimization using deep learning techniques and reviews different examples of deep neural models in non-Euclidean domains; Sect. 3 introduces how triangle meshes are defined in order to represent their features for training neural networks; Sect. 4 explains the architecture of the graph variational autoencoder; Sect. 5 presents the optimization strategy followed and the different stages of the framework; Sect. 6 includes all the optimization results of the two problems; and finally, the last two sections draw some conclusions and discussions about the optimization process applied and the obtained results.

2 Related work

2.1 DL for ASO

ASO consists in finding the design variables that minimize the drag coefficient C_D while minimizing, maximizing, or constraining the lift coefficient C_L (Grey and Constantine 2018) or the mass flow $\dot{m}$ (Corbera et al. 2016) depending on the application. These design variables are usually a set of points along a smooth curve, surface, or bounding box. Common methods for parameterizing aerodynamic shapes include B-splines and Bézier curves (Lepine et al. 2001; Rajnarayan et al. 2018; Rogalsky et al. 1999), FFD (Kenway and Martins 2017), class-shape transformations (Kulfan 2008), PARSEC (Sobieczky 1999), and Bézier-PARSEC (Derksen and Rogalsky 2010), among others (Masters et al. 2017). Unfortunately, convergence and computational costs in ASO are greatly affected by the representation capacity and compactness of the design space. Researchers have found that tens of design variables are required for 2D airfoils (He et al. 2019) and hundreds of shape design variables are required for 3D wing problems (Lyu et al. 2015). This is one of the main issues that engineers find when performing ASO directly dealing with parametric geometries or when relying on intermediate representations that do not directly work with the mesh itself but need a lot of parameters to control the deformation.

To counter this, previous studies have focused on reducing the design dimensionality space and identifying intrinsic data structure while sacrificing minimal accuracy and

information (Hou and Behdinan 2022). Early methods tried to identify the most important directions with respect to the change of response but required too many simulations. Alternatively, not response-based methods such as principal component analysis (Berguin and Mavris 2014; Qiu et al. 2018) or singular value decomposition (Allen et al. 2018; Poole et al. 2019) were proposed. Although these linear models provide optimal solutions for simple cases, real-world cases require non-linear approaches so that they can use the least dimensions to retain similar variance in the data. Topographic mapping (Viswanath et al. 2011, 2014) can solve this problem to some extent, but is still limited to the assumption that data must follow a Gaussian mixture distribution.

In order to get compact representations and capture tiny variations, a vast amount of research on dimensionality reduction has been conducted in the field of deep learning. Deep neural networks such as variational autoencoders (VAEs) (Kingma and Welling 2013) and generative adversarial networks (GANs) (Goodfellow et al. 2014) have successfully represented data from complex high-dimensional distributions, such as images or sounds, by using low-dimensional latent variables. The application of these techniques to ASO is mainly focused on reducing 2D airfoil design variables. For example, the work of Chen et al. (2019) uses the BézierGAN approach (Chen and Fuge 2018) to reduce the dimensionality of the optimization problem by learning from shape variations in the UIUC airfoil database. They are able to learn directly the distribution of points along the curves instead of curve parameters while generating smooth aerodynamic shapes. Choosing the same airfoil database, Achour et al. (2020) adapt a conditional GAN for ASO that can generate airfoil shapes for a given lift-to-drag ratio and area.

For 3D cases, dimensionality reduction is usually applied in two ways. On the one hand, geometries are built by stacking multiple profiles. In particular, Li et al. (2020) propose a new sampling method for airfoils and wings based on a deep convolutional GAN. On the other hand, some authors address the problem of deriving a compact representation for the 3D airfoil as a whole by incorporating a FFD layer into the deep generative model (Chen and Ramamurthy 2021; Li and Zhang 2021). However, none of them directly work with a dataset of 3D meshes without externally parameterizing their possible paths of deformations or without relying on intermediate representations.

2.2 DL for 3D meshes

Deep learning models have been particularly successful when dealing with signals such as speeches, images, or videos in which there is an underlying Euclidean structure. Currently, there is a growing interest in trying to apply deep learning in non-Euclidean geometric data. However, its nature implies that there are no familiar properties such as a common system of coordinates, global parameterization, vector space structure, or shift-invariance. Consequently, basic operations like convolution that are taken for granted in the Euclidean case are not well-defined on non-Euclidean domains (Bronstein et al. 2017).

Such kind of data arises in computer vision and graphics applications when working with 3D representations. For this reason, recent works tried to obtain descriptive and compact representations of 3D shapes for tasks such as generation, interpolation, deformation, segmentation, classification and reconstruction (Bouritsas et al. 2019). Most of the approaches in this direction rely on intermediate representations of 3D shapes such as voxels (Maturana and Scherer 2015; Wu et al. 2016; Brock et al. 2016), point clouds (Achlioptas et al. 2018; Qi et al. 2017; Guo et al. 2020) or mappings to a flat domain (Tan et al. 2018a; Ben-Hamu et al. 2018; Moschoglou et al. 2020), instead of directly working with surface representations like meshes or graphs. Despite the success of such techniques, there are some drawbacks that limit their application to real-world components (Bouritsas et al. 2019): (1) volumetric convolutional neural networks (CNNs) that act on 3D voxels suffer from high computational cost, which implies that they are only suitable for low-resolution 3D volumes; (2) although point clouds are a simple and lightweight alternative to volumetric representation, they are not popular for realistic and high-quality 3D geometry generation due to their lack of an underlying smooth structure; (3) usually highly complex pre and post-processing steps are needed in order to obtain the output surface model, which put a brake on inference.

Learning directly on triangular meshes was only recently explored due to their complex and irregular connectivity. For example, the work of Litany et al. (2018) employs a variational autoencoder with graph convolutional operations that learns a latent space to complete realistic shapes, while both Wang et al. (2018) and Kolotouros et al. (2019) reconstruct 3D triangular meshes from single input images by deforming an initial or template mesh using graph-based convolutional neural networks. Similarly, Wang et al. (2019) propose an end-to-end network to predict 3D deformation that can even reconstruct point clouds. On the other hand, some approaches based on graph convolutional mesh autoencoders try to exploit the learning of deformation properties of objects directly from data, which is already suitable for mesh compression and representation learning. For this purpose, Tan et al. (2018b) use a convolutional autoencoder to extract localized deformation components from mesh datasets with large-scale deformations. The Convolutional Mesh Autoencoder (CoMA) of Ranjan et al. (2018) allows to model and sample stronger deformations on

high-resolution triangle meshes compared to previous methods, even supporting asymmetric facial expressions. This is one of the most outstanding works in mesh sampling and it was used by Yuan et al. (2020) to design a pooling operation for meshes that avoid the generation of highly irregular triangles during mesh simplification by keeping track of the mapping between coarser and finer meshes. Also following the CoMA architecture, Tretschk et al. (2020) added a novel embedded deformation layer that explicitly models point displacements of non-rigid deformations in a lower dimensional space. This serves as a local rigidity regularizer and it allows us to achieve higher-quality autoencoding results compared to several baselines and existing approaches.

Similar to CoMA, our GVAE architecture employs spectral graph convolutions and opts for an autoencoder to create a generative model. However, the following weak points prevent the model ideated by Ranjan et al. (2018) from being directly used in the proposed optimization framework:

- It was not designed to work with generical datasets in which parts have different features in more than a local region or even different topology, that is, mesh connection properties.
- Spectral graph convolutions cannot guarantee the preservation of the signal smoothness, which increases the probability of finding artifacts and surface irregularities in the generated geometry.
- Chebyshev convolutions are intended to capture localized information using spectral filters, but they are not good at all capturing global patterns by aggregating information from different local regions.

As a solution to these issues, two new features have been incorporated:

- A pre-processing mesh topology homogenization step which makes the meshes of the dataset more suitable for graph neural networks guarantees the numerical stability of the optimization algorithm, and speeds up the convergence. Thereby, the user can directly perform optimizations in their own datasets even if the quantity of geometries is very low.
- The use of spline-based graph convolutions in the encoding phase. In contrast to related approaches that filter in the spectral domain, spline-based convolutions purely aggregate features in the spatial domain. It increases numerical stability, provides flexibility, guarantees the balance between localized and global information, and enables to capture and represent smooth variations in signals across the graph in a way that respects the local structure.

3 Mesh representation

One of the most common types of CAD files used to describe the shape and geometry of a 3D object without a fully parametric way is the *.stl format. Using a triangular mesh-based approach, it describes the surface of any object through the coordinates of the vertices that define every triangular facet, together with the components associated with its perpendicular vector that points outwards (normal vector). The proposed framework is conceived to interact with these mesh-based files due to its simplicity and widespread use in 3D model libraries such as *GrabCAD* or *3D CAD Browser*. Nevertheless, it is also compatible with others like *.ply and *.obj that follow the same structure but adding more information.

Aiming to homogenize different file formats, the information contained is transformed into a Face-Vertex structure (Jabón et al. 2023) so that triangular meshes are defined as $\mathcal{M} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$. The vertices matrix $\mathcal{V} \in \mathbb{R}^{nx3}$ contains the three-dimensional Cartesian coordinates of all the nodes, the facets matrix $\mathcal{F} \in \mathbb{N}^{mx3}$ includes the labeled vertices that form each facet, and the edges matrix $\mathcal{E} \in \mathbb{N}^{2xl}$ represents the connectivity between all vertices. Basically, the latest describes the adjacency matrix of the mesh in COO format (coordinates), which is needed to define the mesh in Pytorch Geometric (Paszke et al. 2019; Fey and Lenssen 2019) as an undirected graph. However, it is necessary to carry out the following two processes to make the meshes of the dataset more suitable for graph neural networks, to guarantee the numerical stability of the optimization algorithm, and to speed up the convergence.

On the one hand, node features are normalized by centering the mesh at (0,0,0) and scaling it to fit a cube with unitary edges. Consequently, the initial vertices matrix $\mathcal{V}$ is transformed into $\mathcal{V}'$. This step plays a crucial role in leveling the playing field for data, thereby enabling models to learn effectively and deliver accurate results (not sub-optimal solutions). Furthermore, it is ensured that the order/range of the input data and the weight/bias matrices that will be initialized at a later stage is practically the same, avoiding the exploding/vanishing gradient problem.

On the other hand, all meshes must have the same connectivity, that is, all meshes must be represented with the same graph connections and number of nodes but different node properties, which corresponds to Cartesian coordinates of the vertices. In order to achieve this goal, a reference mesh $\mathbf{M}_0$ with n vertices, $\mathbf{F}_0$ facets definition and $\mathbf{E}_0$ adjacency matrix is selected to be deformed to get the same shape as the target normalized mesh $\mathcal{M}'$. This is a common problem in the literature that is currently implemented in Pytorch 3D (Paszke et al. 2019; Ravi et al. 2020). The approach learns to offset the vertices of

the initial mesh, $\mathbf{O} \in \mathbb{R}^{nx3}$, such that the predicted mesh is closer to the target mesh at each optimization step while keeping the same connectivity:

$$\mathbf{V} = \mathbf{V}_0 + \mathbf{O} \mid \mathbf{V} \approx \mathcal{V}$$

In order to make the problem presented trainable, a loss function that measures the similarity between the initial and target mesh is needed. Since 3D shapes are defined by vertices and facets whose accurate topological similarity is difficult to measure, a set of points are uniformly sampled from the surface of each mesh and the chamfer loss is used to constrain the location of the vertices (Fan et al. 2017). Nevertheless, solely minimizing the chamfer distance L_c between the predicted and the target mesh would lead to a non-smooth shape. For this reason, smoothness is enforced by adding shape regularizers as stated in Wang et al. (2018, 2019): (1) the edge length loss L_c to penalize flying vertices, which usually cause long edges; (2) the normal loss L_n to enforce consistency across the normals of neighboring faces; (3) the laplacian loss L_n to encourage neighboring vertices to have the same deformation movement and prevent self-intersection. Therefore, the optimizer (eg. SGD, RMSprop, Adagrad, Adam, etc) minimizes the following weighted loss function to guarantee appealing deformation results:

$$L = \eta_1 L_c + \eta_2 L_e + \eta_3 L_n + \eta_4 L_l$$

In short, given a dataset of triangle-based mesh files (*.stl, *.ply or *.obj) in which each mesh is universally defined as $\mathcal{M}_i = (\mathcal{V}_i, \mathcal{F}_i, \mathcal{E}_i)$, the pre-processing step needed to adapt all the meshes to be used in the GVAE architecture involves the use of the following functions:

$$\left.\begin{array}{l} f_1 : \mathcal{V}_i \longrightarrow \mathbf{V}_i \\ f_2 : \mathcal{F}_i \longrightarrow \mathbf{F}_0 \\ f_3 : \mathcal{E}_i \longrightarrow \mathbf{E}_0 \end{array}\right\} \mathcal{M}_i \longrightarrow \mathbf{M}_i,$$

where $\mathcal{V}_i \in \mathbb{R}^{n_i x3}$, $\mathbf{V}_i \in \mathbb{R}^{nx3}$, $\mathcal{F}_i \in \mathbb{N}^{m_i x3}$, $\mathbf{F}_0 \in \mathbb{N}^{mx3}$, $\mathcal{E}_i \in \mathbb{N}^{2xl_i}$ and $\mathbf{E}_0 \in \mathbb{N}^{2xl}$.

Therefore, each mesh is represented as $\mathbf{M}_i = (\mathbf{V}_i, \mathbf{F}_0, \mathbf{E}_0)$. This implies that both the facets and edges matrices are shared by all the elements of the dataset, while the vertices matrix that contains the centered and normalized coordinates of all the nodes of the mesh is the unique differentiating attribute. The flow of actions of the pre-processing step is depicted in Fig. 2.

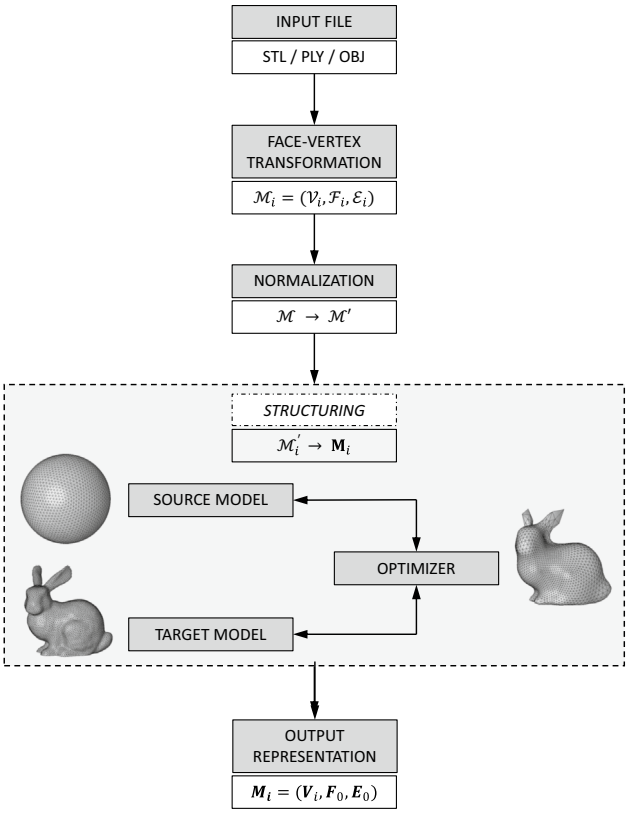

Fig. 2 General workflow of the pre-processing step needed to normalize the vertices position and modify the topological structure of all the meshes of the dataset

4 Mesh autoencoder

Generative modeling consists in concisely capturing the characteristics of large datasets in order to sample synthetic data that resemble the training data distribution. As presented in Sect. 2, two major families stand out: VAEs (Kingma and Welling 2013) and GANs (Goodfellow et al. 2014). The present work focuses on the first approach due to their remarkable capability to generate near-original and completely new content with a low-dimensionality vector. This condensed vector will subsequently be employed by a different agent to guide the optimization process.

In a nutshell, VAEs are autoencoders whose encoding distributions are regularized during training through the use of Variational Bayesian methods in order to ensure that the latent space has good properties, allowing the generation of new data based on the correlation of trained features. Following the work of Ranjan et al. (2018), the proposed GVAE structure models geometric variations of 3D objects using CNNs directly applied to triangle-based meshes. In this chapter, the graph convolution operators and the mesh encoding-decoding architecture employed are presented, as well as some implementation details.

4.1 Graph convolutions

To form a complete neural network architecture, the proposed model adopts spline-based (Fey et al. 2017) and

Chebyshev spectral (Defferrard et al. 2016) graph convolutional operators to, respectively, construct the encoder and decoder.

In order to generalize convolutional neural networks for graph inputs, a large number of approaches based on spectral graph theory have been proposed. Let $\mathcal{L} = \mathcal{D} - \mathcal{W}$ denote the non-normalized graph Laplacian and $\mathcal{L} = 1 - \mathcal{D}^{-1/2}\mathcal{W}\mathcal{D}^{-1/2}$ the normalized one, where $\mathcal{W}$ and $\mathcal{D}$ are the adjacency and degree matrices, respectively. As it is a real symmetry matrix, it is diagonalized by the Fourier basis $\mathcal{L} = \mathcal{U}\mathcal{A}\mathcal{U}^T$, where $\mathcal{U}$ is created with the orthogonal eigenvectors and $\mathcal{A}$ is the diagonal matrix with non-real negative eigenvalues. Following this, the graph convolution is defined in the Fourier domain as

$$x * y = \mathcal{U} g_\theta \mathcal{U}^T x,$$

where g_θ is a filter that depends on $\mathcal{A}$.

Defferrard et al (2016) approximate this filter using Chebyshev polynomials of order K, providing a more efficient filtering algorithm than previous works (Bruna et al 2013; Henaff et al 2015), whose kernel size determines the range of aggregated local K-neighborhoods:

$$g_\theta = \sum_{k=0}^{K-1} \theta_k T_k(\mathcal{A}),$$

where $\theta \in \mathcal{R}^K$ is a vector of Chebyshev coefficients and $T_k \in \mathcal{R}^{nxn}$ is the Chebyshev polynomial of order k such that $T_0 = 1, T_1 = x$, and $T_k = 2xT_{k-1}(x) - T_{k-2}(x)$.

Scaling the Laplacian with $\tilde{\mathcal{L}} = 2\mathcal{L}/\lambda_{max} - I$ in order to have all eigenvalues lying in [-1,1], the spectral graph convolutions used in the decoder are defined as

$$x * y = \mathcal{U} \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathcal{A}}) \mathcal{U}^T x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathcal{L}}) x.$$

On the other hand, the encoder employs a convolution operator based on B-splines, which makes the computation time independent of the kernel size. In contrast to the approaches that filter in the spectral domain, this method aggregates features purely in the spatial domain by using continuous kernel functions parameterized by a fixed number of trainable weights. The spline-based convolutional layer receives as input irregularly structured data that is mapped to a graph following the steps presented in Sect. 3. However, unlike the spectral approach, edge features must be extracted from node features, which can be tackled by saving the relative Cartesian coordinates of linked nodes in $\mathbf{W}$. Note that $\mathbf{U} \in \mathbb{R}^{nxnx3}$ contains the normalized Cartesian coordinates $\mathbf{u}_{i,j} \in [0, 1]^3$ of the edge that connects each pair of nodes, being null if they are not connected.

Let $((N^m_{1,i})_{1<i<k_1}, \ldots, (N^m_{d,i})_{1<i<k_d})$ denote open B-spline bases of degree m that are based on uniform and equidistant knot vectors (Piegl and Tiller 1997), with $k = (k_1, \ldots, k_d)$ defining the d-dimensional kernel size. The continuous kernel function proposed by Fey et al. (2017) is parameterized by a trainable parameter $\mathbf{w}_{p,q} \in \mathbf{W}$ for each element p resulted from the Cartesian product $\mathcal{P} = (N^m_{1,i})_i \times \ldots \times (N^m_{d,i})_i$ in each of the dimension q_{dim} of the node features vector. Therefore, the continuous convolutional kernel is determined by

$$g_q(\mathbf{u}) = \sum_{\mathbf{p} \in \mathcal{P}} \mathbf{w}_{p,q} \cdot B_p(\mathbf{u}) = \sum_{\mathbf{p} \in \mathcal{P}} \mathbf{w}_{p,q} \cdot \prod_{i=1}^{d} N^m_{1,p_i}(\mathbf{u}_i)$$

Given the kernel functions $g = (g_1, \ldots, g_q)$, the spatial convolution operator is defined for each node i as follows:

$$(x * y)(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{q=1}^{d_{dim}} \sum_{j \in \mathcal{N}(i)} x \cdot g_q(\mathbf{u}_{i,j}),$$

being $\mathcal{N}(i)$ the neighborhood set of node i.

4.2 GVAE architecture

Following the generalization of traditional VAEs to graphs through the use of the previously defined convolutional operators, the structure of the network proposed is illustrated in Fig. 3.

The model basically aims to find a probabilistic encoder and a probabilistic decoder. While the encoder maps the posterior distribution from the mesh $\mathbf{M}_i$ to the latent vector $\mathbf{z}_i$, the decoder produces a plausible corresponding mesh $\tilde{\mathbf{M}}_i$ from the mentioned latent vector. The encoder of our network consists of four spline-based convolutional layers of degree $d = 2$, each of them followed by a graph normalization (Cai et al. 2020) and a non-linear ReLU activation layer. The output of the last convolutional layer is mapped to a mean vector and a deviation vector with the dimension of the latent space using two different fully connected layers. Then, the reparametrization trick (Kingma and Welling 2013) is applied in order to make the network suitable for backpropagation. The decoder similarly consists of a fully connected layer that firstly transforms the latent vector from $\mathbb{R}^\gamma$ to $\mathbb{R}^{\alpha x \beta}$, but employing different convolutional layers. There is no doubt that spectral approaches are less accurate for mapping the input geometrical relations since the information is only encoded based on the connectivity and node features; however, information about edge attributes cannot be provided if we want to reconstruct a mesh from a latent vector and we only know the edge connectivity of the output. Therefore, the decoder employs four spectral convolutional layers with Chebyshev polynomials filter of size $K = 6$. All of them except the last one are followed by graph normalization and non-linear Leaky ReLU activation layers.

In order to train the GVAE network, the squared error (SE), also known as the squared L_2 norm measured as the sum over

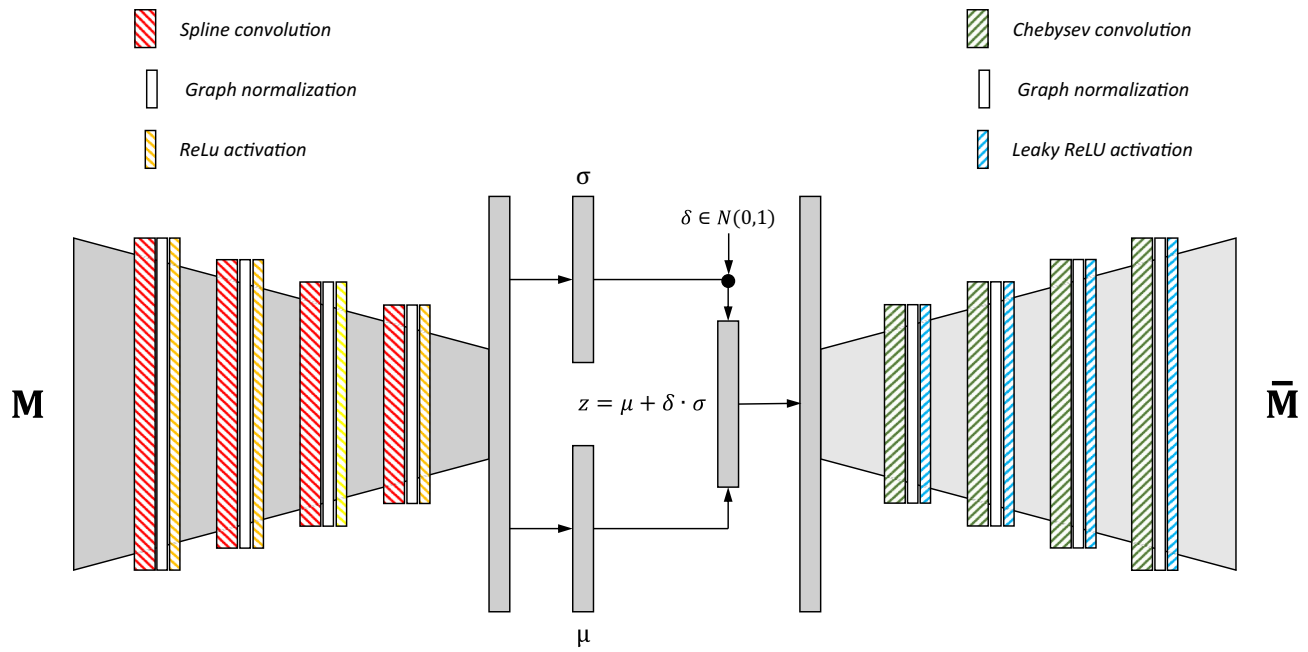

Fig. 3 GVAE structure. The mesh is firstly encoded using Spline convolutional layers, graph normalization, and ReLU activation functions. Then, two fully connected layers generate the latent vector through the reparametrization trick. Finally, the decoder consists of Chebyshev convolutional layers, graph normalization, and Leaky ReLU activation functions

data points, is used as the reconstruction loss. Combined with the KL-divergence (Kullback and Leibler 1951) to make the distribution of the encoder output as close as possible to a standard multivariate normal distribution, the loss function for each mesh i of the dataset is defined as follows:

$$L_i = \sum_{j=1}^{n} \|\tilde{\mathbf{v}}_i^j - \mathbf{v}_i^j\|_2^2 + \epsilon D_{KL}(q(\mathbf{z}_i)\|p(\mathbf{z}_i)),$$

where $\tilde{\mathbf{v}}_i^j$ and $\mathbf{v}_i^j$ are the output and pre-processed coordinates of each vertex j of each mesh i, $q(\mathbf{z}_i) = N(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$ is the posterior or encoding distribution, $p(\mathbf{z}_i) = \mathrm{N}(0,1)$ is the prior distribution, and ϵ is a parameter used to adjust the priority between the reconstruction loss and KL-divergence (Asperti and Trentin 2020). Since $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are the mean and variance vectors of the Gaussian distributions from which the latent vector $\mathbf{z}_i$ that encodes each mesh i is sampled, the KL-divergence is defined as

$$D_{KL}(q(\mathbf{z}_i)\|p(\mathbf{z}_i)) = -\frac{1}{2}\sum_{k=1}^{\gamma}(1 + log(\sigma_{ik}^2) - \mu_{ik}^2 - \sigma_{ik}^2),$$

where γ is the dimension of the latent space. Therefore, the total loss function for the model is calculated as

$$L = \frac{1}{m}\sum_{i=1}^{m} L_i,$$

where m is the total dimension of the dataset.

4.3 Implementation details

By default, the dimension of the latent space is set to 4. The encoder takes the 3-dimensional node features and performs Spline convolutions with the following size transformations: 3-4-8-4-4. Likewise, the decoder takes the output from the fully connected layer and, once reshaped, performs the following Chebyshev convolutions to get the reconstructed 3-dimensional features: 4-4-8-4-3. All neural networks' weights and biases are initialized using a normal distribution with 0 mean and 0.1 standard deviation. However, the user can directly adjust these hyperparameters if needed for other applications that employs larger or more complex datasets.

The proposed GVAE network has been implemented in Pytorch Geometric (Paszke et al. 2019; Fey and Lenssen 2019), selecting as optimizer the well-known AdamW optimizer (Kingma and Ba 2014; Loshchilov and Hutter 2018). More details about the number of epochs and the learning rate can be found inside each specific application.

5 Optimization approach: coupling GVAE and GA

Once the network is trained and all meshes of the dataset are embedded and properly distributed along the latent space, it is time to go through it to generate both initial meshes and new shapes not seen during training. However, it is not just

a matter of sampling random latent vectors. The search for the most suitable geometry that maximize and/or minimize certain aerodynamic coefficients requires the use of an intelligent agent. We are basically trying to solve an ASO problem given a low-dimensional design space in which the aerodynamic performance is calculated by a CFD solver since really complex shapes are generated during the process. All of that leads to the basics of a multi-objective optimization problem, which is formulated as follows:

$$\left.\begin{array}{l} minf_1(\mathbf{x}) = C_D \\ maxf_2(\mathbf{x}) = C_L \end{array}\right\} MOOP,$$

where C_L and C_D are the lift and drag coefficients, while $\mathbf{x} \in S$ represents the design variables that belong to the design (latent) space of dimension γ, being upper and lower bounds established using the maximum and minimum values from reconstructed latent vectors plus 25 %. However, note that depending on the application, an alternative problem could be formulated as follows:

$$\left.\begin{array}{l} minf_1(\mathbf{x}) = C_D \\ minf_2(\mathbf{x}) = C_L \end{array}\right\} MOOP,$$

Among the wide range of existing methods to address MOOP, the authors consider genetic algorithms (GAs) the most suitable technique for this problem due to the fact that they do not require sensitive information nor access to the equations of the solver. Therefore, heuristic algorithms, instead of gradient-based techniques, are positioned as the most appropriate option when working with commercial software. In particular, the NSGA-II (Deb et al. 2002) is chosen as the optimization agent in charge of handling the geometry creation by defining which latent representations are going to be decoded. This algorithm is characterized by the use of an explicit diversity-preserving strategy in combination with an elite preservation approach. Both features are based on the *Pareto Dominance*, a powerful approach used when the objectives stated are in conflict with each other.

The optimization subroutine governed by the *NSGA-II* integrates (1) the trained GVAE network that is able to embed (encode) all the training dataset into a compact design space and generate (decode) different geometries in *.stl format based on determinate latent vector values; (2) the connection with *ANSYS Spaceclaim* to transform the *.stl file into a solid and to create the fluid control volume where boundary conditions are assigned; (3) the connection with *ANSYS Meshing* to update, mesh the fluid domain and evaluate its maximum skewness, which must be kept below 0.95 to avoid convergence difficulties and guarantee an accuracy solution; (4) the connection with *ANSYS Fluent* in order to import the mesh in the CFD model and analyze the aerodynamic performance. Therefore, the first three points are associated with candidates' generation through the definition of the design variables by the GA, while the fourth one corresponds to candidates' fitness evaluation. At the end of the optimization process, the optimal Pareto front with all generated geometries is made available so that the designer can select the best solution for a given trade-off between the different objectives (drag and lift coefficients). Following this, Fig. 4 depicts the overall optimization workflow of the proposed approach.

In summary, the GVAE is in charge of generating new geometries, while the NSGA-II is the intelligent agent that controls the optimization process by deciding which geometries to generate. Therefore, a variation in the problem formulation only affects the optimization agent and not the generative one: if geometrical or performance restrictions had to be considered, it would simply be necessary to add a geometrical evaluation subroutine or an objective function penalty to limit the exploration of a determinate region of the latent space. This is a good feature that results from the combination of both algorithms. It makes the framework very versatile and allows the designing of a much more sophisticated geometry generator than current shape algorithms.

It is important to note that in most optimizations for industrial applications, each analysis takes much longer (minutes or even hours) than the data processing of the GAs (milliseconds). This fact highlights the importance of selecting a correct stopping criterion for the optimization (Corbera et al. 2016). In MOOP, the stop criterion is based on two performance metrics (Deb 2011): the convergence to the global optimal and the diversity of the generated solutions. The GAs require a large number of simulations to converge to the global Pareto optimal front, which could mean a potential drawback for engineering problems that need long simulation times. Initializing the optimization with a reasonable number of candidates that are properly distributed is a method particularly attractive and extended in the industry, although it does not guarantee the Pareto optimal front. If the convergence is not achieved after the initial run, the optimization is iteratively restarted by adding a few generations at a time considering the performance metrics (convergence and diversity). The default algorithm setup for the optimization is the following: a population size of 15 individuals that evolve throughout 10 generations.

Finally, it must be considered that the effectiveness of a GA is also subject to the appropriate definition of the design variables and the crossover/mutation strategy. They should be consistent with the kind of data to be processed (Sigmund and Maute 2013) because a wrong decision could lead to a lack of exploration capability or premature convergence. For this approach, a tournament selection of size 3 is used and a uniform crossover strategy is adopted with a probability equal to 0.6. Furthermore, following typical values in the literature, a value equal to 0.01 is applied as the mutation rate.

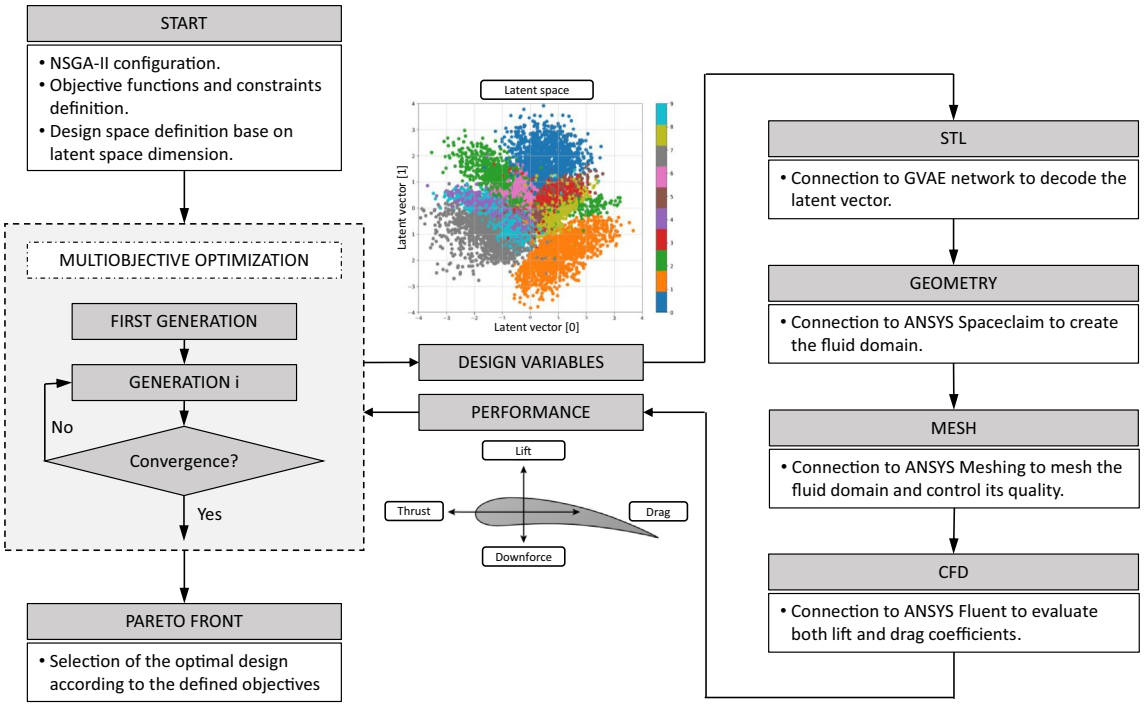

Fig. 4 Overall optimization workflow. Both the design and the evaluation processes are embedded within the optimization process, being used by the NSAG-II to create new meshes and evaluate every candidate in each generation. This connection between the optimization engine, the design, and the evaluation process is repeated recursively until the criteria of convergence is satisfied. Finally, a Pareto front is obtained and the optimum design is selected according to the defined goals

6 Application

In this section, the proposed approach is examined to test the capability to generate new models and to demonstrate its usefulness when optimizing two datasets. The first one is used as a "case study" to validate the algorithm, while the second one shows the application of the method to a real-world optimization problem. Rather than focusing on getting a particular numerical result with a highly accurate CFD model that can replicate wind-tunnel experiments, the target of this optimization workflow is to search for the optimal design within all the existing solutions in the dataset. We demonstrate that the proposed GAVE architecture is able to learn realistic shape variation between different geometries that are embedded in the latent space, being this latent space the design space that the *NSGA-II* algorithm uses to generate new geometries with interpolated features. To accomplish this task, the following computer with CUDA compatibility was used: Intel Core i7-11800 H/32GB 3200MHz/1TB SSD/RTX 3080.

Both optimizations are carried out using the shear-stress transport (SST) k-ω model. This hybrid two-equation eddy-viscosity turbulence model is used for many aerodynamic applications, thanks to its robustness and reasonable accuracy. It gradually changes from the standard k-ω model in the inner region of the boundary layer to a high-Reynolds-number version of the k-ϵ model in the outer part of the boundary layer. Furthermore, in order to increase the efficiency of the model by reducing the number of elements employed and, hence, accelerating the optimization, a polyhedral mesh is employed. It is adjusted to have a size of 0.004 m around the geometry studied, with five inflation layers to capture the boundary layer region. This leads to a total number of around 3e6 and 9e6 polyhedral elements in each geometry of the first and second datasets, respectively. In turn, the CFD analysis of the second dataset uses a body of influence in order to locally refine the mesh for capturing the wake.

On the other hand, regarding the boundary conditions, it is assumed that the three-dimensional incompressible flow is in a steady state, having the fluid volume a length of 3 L upstream and 9 L downstream. The working fluid is air ($\rho = 1.225\ kg/m^3$, $\mu = 1.7894\text{e}{-}05\ kg/m^{-1}s^{-1}$) and it has been set an inlet speed of 30 m/s and an outlet gauge pressure of 0 Pa. For the rest of the faces of the enclosure box, it is assumed a symmetry condition two by two, while the face of the studied geometry is considered as a stationary wall.

6.1 3D airfoils

Since there are a lot of different airfoils that can be used in different scenarios inside the aerospace industry and the

nature to produce lift or downforce efficiently, the first dataset consists of seven 3D airfoils in *.stl format. Considering all possible generic shapes that airfoils can take, each one of the profiles included in the dataset belongs to one of the following groups:

- Under-cambered: NACA 6409
- Flat-bottom: Drela AG03
- Semi-symmetrical: NACA 23015.
- Symmetrical: NACA 0008.
- Reflexed: Martin Hepperle MH78.
- Flat: Flat-plate.
- Stepped: Kline-Fogleman KFm-2.

All these training data are depicted in Fig. 5 as input of the first block, where all the 3D airfoils are pre-processed to make them suitable for the GVAE architecture. Therefore, as previously explained, it is normalized and topologically homogenized by deforming a parallelepiped with 2803 vertices and 5602 triangles. In order to achieve this, the "mesh transformation" module learns the offset applied to each vertex of the source mesh such that the predicted mesh is closer to the target mesh at each optimization step. To this effect, the following weights are applied to the loss function that measures the deviation between meshes and includes various shape regularizers: $\eta_1 = 1.0, \eta_2 = 0.25, \eta_3 = 0.05$, and $\eta_4 = 0.5$. Using the Adam optimizer with a learning rate of 0.01 that is exponentially decreased during 2000 iterations, an average total loss of $L = 0.00157$ per mesh is achieved. The resulting meshes are shown in Fig. 5 as the output of this first block.

Once the dataset has been treated, the GVAE structure is trained to learn the latent representations of input meshes, which dimension is fixed by the parameter $\gamma = 4$. In order to accomplish this task, the GVAE network is trained for 5000 epochs using the AdamW optimizer with a learning rate of 0.0025 and a weight decay of 0.5 every thousand epochs. This configuration leads to a loss value of $L = 0.554$, a quantity obtained by weighting the reconstruction error and the KL-divergence error with a balance parameter $\epsilon = 0.1$, which values are 0.176 and 0.378, respectively.

Finally, since the desired diversity and convergence are achieved after the initial run following the default setup, 150 different candidates are generated by the *NSAG-II*. This interaction between the optimization engine and the *ANSYS Fluent* to produce each geometry by selecting the proper genotype and to assess each one by calculating its drag and lift coefficients leads to the Pareto front shown Fig. 5. This set of Pareto optimal solutions resulted from the MOOP that aims to reduce drag and increase lift is mixed with the initial dataset geometries (red markers). It can be clearly appreciated how the optimization engine is able to fulfill all the empty gaps between the initial individuals, ensuring a proper distribution between all generated geometries. In order to probe the interpolation and embedding capabilities of the GVAE structure together with the exploration power of the evolutive algorithms, three geometries have been selected due to their excellent properties. Their profiles are shown in Fig. 5 as the output of the optimization block, whereas their drag/lift coefficients and the encoding latent vector values are presented in Table 1.

6.2 Cars' rear wings

The dataset of cars' rear wings is formed by a total of eight 3D geometries in *.stl format. Half of them are built using a customized profile, while the other half is based on the inverted NACA 6412 profile. All of them are created using an angle of attack of 8 degrees, being the main difference between them the way in which the rear wing is fixed to the car and the design of the endplate (if used). The initial dataset and the pre-processed solutions with a normalized and homogenized topology are depicted in Fig. 6 as input of the first and second blocks, respectively. The mesh transformation is carried out by deforming a bride-shaped mesh that envelops all the dataset geometries. This source mesh has been built using 5019 vertices and 10034 triangles and, for deformation training, the following regularizers are used: $\eta_1 = 1.0, \eta_2 = 0.5, \eta_3 = 0.01$, and $\eta_4 = 0.1$. In order to obtain the modified meshes, the Adam optimizer with a learning rate of 0.005 that is decreased linearly was used, achieving an average total loss of $L = 0.00067$ per mesh after 2000 iterations.

These pre-processed meshes are used as the training dataset of the GVAE network with a latent dimension of $\gamma = 4$ and the same configuration of the preceding example. This training is successfully completed with a total loss of $L = 0.349$, a value that results when adding the reconstruction error $L_R = 0.145$ with the weighted KL-divergence error $\epsilon L_{KL} = 0.204$. However, due to the higher complexity of this dataset in comparison with the first one, it has been necessary to increase the training iterations up to 10000 epochs, using the same weight decay as in the standard configuration for the first 9000 epochs; while for the last 1000 epochs, a decay of 0.1 every hundred iterations was employed. Furthermore, the loss balance parameter ϵ has been reduced to 0.05 to guarantee the convergence of the process.

Once the GVAE structure is trained to embed all the input meshes in the latent space, the *NSAG-II* is coupled with the GDL network and *ANSYS Fluent* in order to produce a total of 150 different designs. It is important to highlight that, unlike the previous case, the optimizer is configured to maximize downforce instead of lift. All the different geometries generated during the optimization process are collected in the Pareto front of Fig. 6, using red markers for those geometries of the initial dataset.

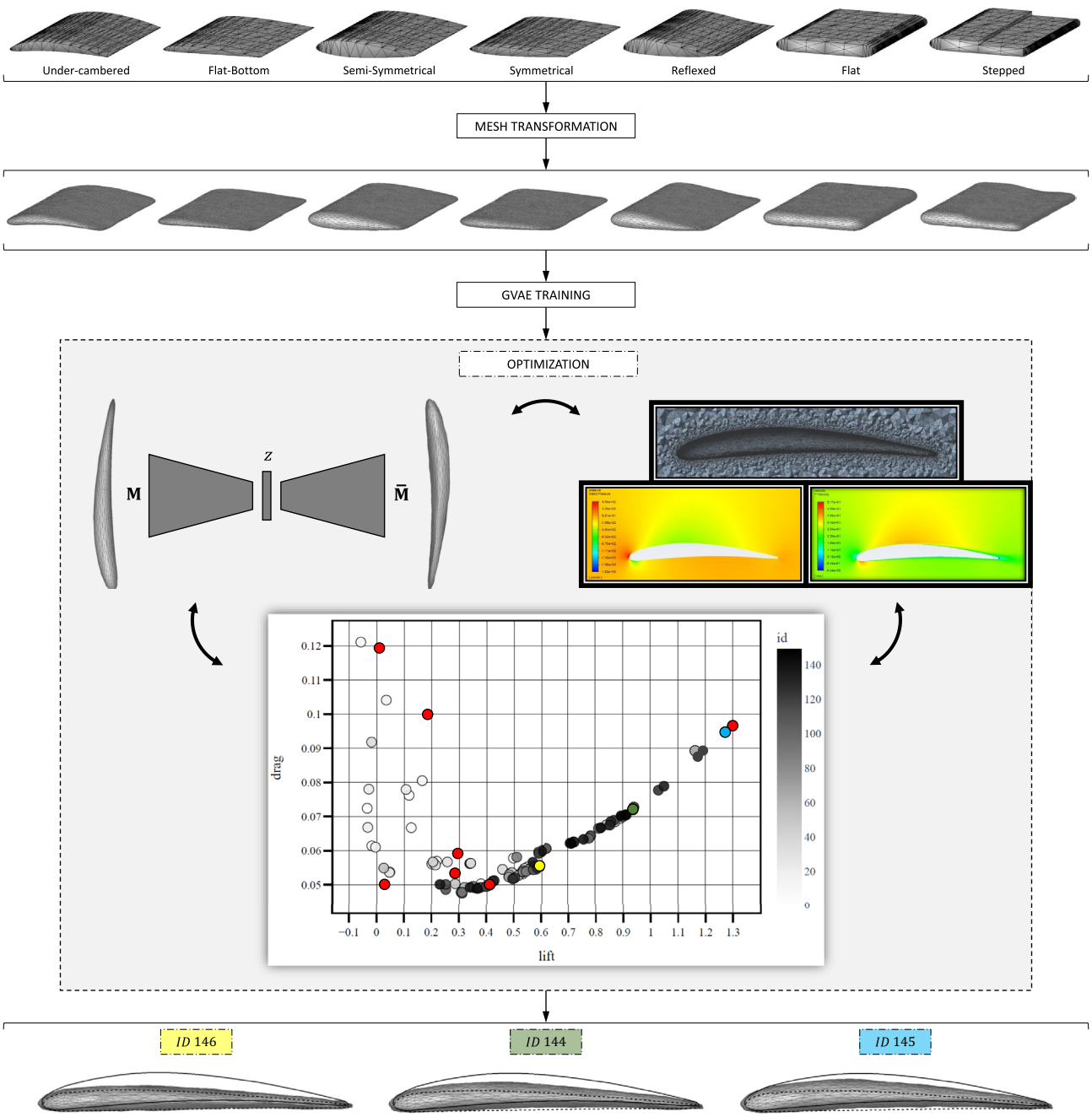

Fig. 5 ASO of the 3D airfoils dataset. Once the input *.stl files are transformed and the GVAE network trained, the optimization process governed by the NSGA-II starts. The resulting Pareto front is presented, being the parts of the initial dataset illustrated with the red points. Finally, three solutions are selected and compared with initial profiles NACA 6409 (under-cambered) and NACA 0008 (symmetrical)

As opposed to the previous case, there are not a lot of empty gaps to fulfill, but rather to improve the properties of the initial geometries with small variations or through the exchange of information between them. As a result of the optimization block, two geometries are depicted because they have improved the properties of the initial wings "NACA model 2" and "NACA model 3," meanwhile another one is shown to probe the interpolation properties of the model, which leads to a geometry resulted from the combination of models 1 and 3. Detailed numerical results are presented in Table 2.

Table 1 Numerical values resulted from the optimation of the first dataset in comparison with the reconstructed parts. Lift and drag coefficients were obtained through ANSYS Fluent, while the latent vector is the encoding vector of each geometry in the GVAE architecture

		Drag	Lift	Latent vector			
Dataset	Under-cambered	0.097	1.299	0.7094	0.8465	− 0.0427	1.7122
	Flat-bottom	0.050	0.425	0.7685	0.7849	− 1.1323	0.6195
	Semi-symmetrical	0.060	0.302	− 1.1150	− 0.0711	− 0.9970	0.8697
	Symmetrical	0.047	0.012	0.3790	− 0.0141	− 1.7082	0.0251
	Reflexed	0.057	0.293	− 0.8843	0.1099	− 0.4012	1.3264
	Flat	0.121	0.025	− 1.2048	0.0123	− 1.2720	− 0.7144
	Stepped	0.100	0.215	− 1.2968	1.2104	− 0.2362	− 0.2985
Sim.	ID 144	0.072	0.937	0.4462	1.4065	− 0.9131	1.4900
	ID 145	0.093	1.278	0.6012	1.1325	− 0.8765	1.2345
	ID 146	0.055	0.595	0.7186	1.3790	− 1.5251	1.4647

7 Discussion

The main objective of the proposed framework is to perform aerodynamic shape optimizations given a dataset of geometries modeled by unstructured triangle meshes. To that end, it employs a pre-processing step in which meshes are normalized and topologically homogenized, a graph neural network in the form of a variational autoencoder for representation learning, and an evolutive optimization agent in charge of generating solutions by sampling from the latent space and evaluating all of them through the connection with the CFD software.

In this section, we look deep into the results obtained when the automatic framework is applied to the 3D airfoils and the 3D car's rear wings datasets. Whereas the first one can be treated as a case study employed to validate the approach due to its geometrical simplicity and previous use by other researchers, the second one is the perfect example of a real-world engineering problem application. For this reason, following the results unveiled in Sect. 6, some interesting discussions can be drawn:

- The combination of the pre-processing technique and the GVAE network produces really encouraging and accurate results for shape compression and high-quality reconstruction of the input geometries. The reconstruction loss values obtained by the graph neural network are practically negligible, which demonstrates the power of the combination between the spline-based and the spectral convolutional layers for the task of representation learning of 3D objects treated as graphs. In other words, the proposed architecture has been validated when embedding all these complex 3D triangle meshes into low-dimensional latent space, being able to successfully capture all the deformation patterns and variations in the dataset. Hence, it allows to encode and decode the initial samples of the dataset with minimal error, as well as generating new geometries not seen during training. However, although the proposed framework can reduce the data dimensionality and identify intrinsic data structure while sacrificing minimal accuracy and information, the use of the deformation technique presented in Sect. 4 to make the meshes of the dataset suitable for the graph neural network leads to solutions with a smoother surface than the target ones. This is due to the use of the Laplacian loss and it implies that 90-degree flat edges are not directly preserved but rounded. This phenomenon is appreciable in Fig. 3 when trying to reconstruct a bunny and it is minimized when reducing the Laplacian loss to a minimum in which self-intersection does not appear.
- While the proposed GVAE structure excels when generating new parts through interpolation, the extrapolation capabilities are quite limited. Taking a deeper look into the presented Pareto fronts, 95 % of the candidates generated during the evolutionary process remain inside the maximum and minimum lift and drag values of the initial dataset. Only a few like the ID 97 and ID 120 from the second dataset are able to increase the performance of the initial best geometry by creating a profile that is smaller than the two input profiles (NACA 6412 and the custom one). The easiest way to generate a geometry better than the NACA model 3 would have been to delete the support and use only the profile area or just a little bit of support acting like an endplate. However, the model has not been able to create such a solution. So as to understand why this event took place, we decoded latent representations that reside outside the segment defined between to samples by setting $a \in (-\infty, 0) \cup (1, \infty)$. Unfortunately, we realized that most of the generated geometries outside this segment lead to self-intersecting surfaces, which are discarded during the evolutionary process as a CFD protection measure. This fact could be addressed in future versions including mesh post-processing steps, which undoubtedly would improve the optimization results by expanding the feasible design space and increasing the probability of finding the desired global minimum.
- During the optimization process of the first and second datasets, 19 and 16 % of the parts were, respectively,

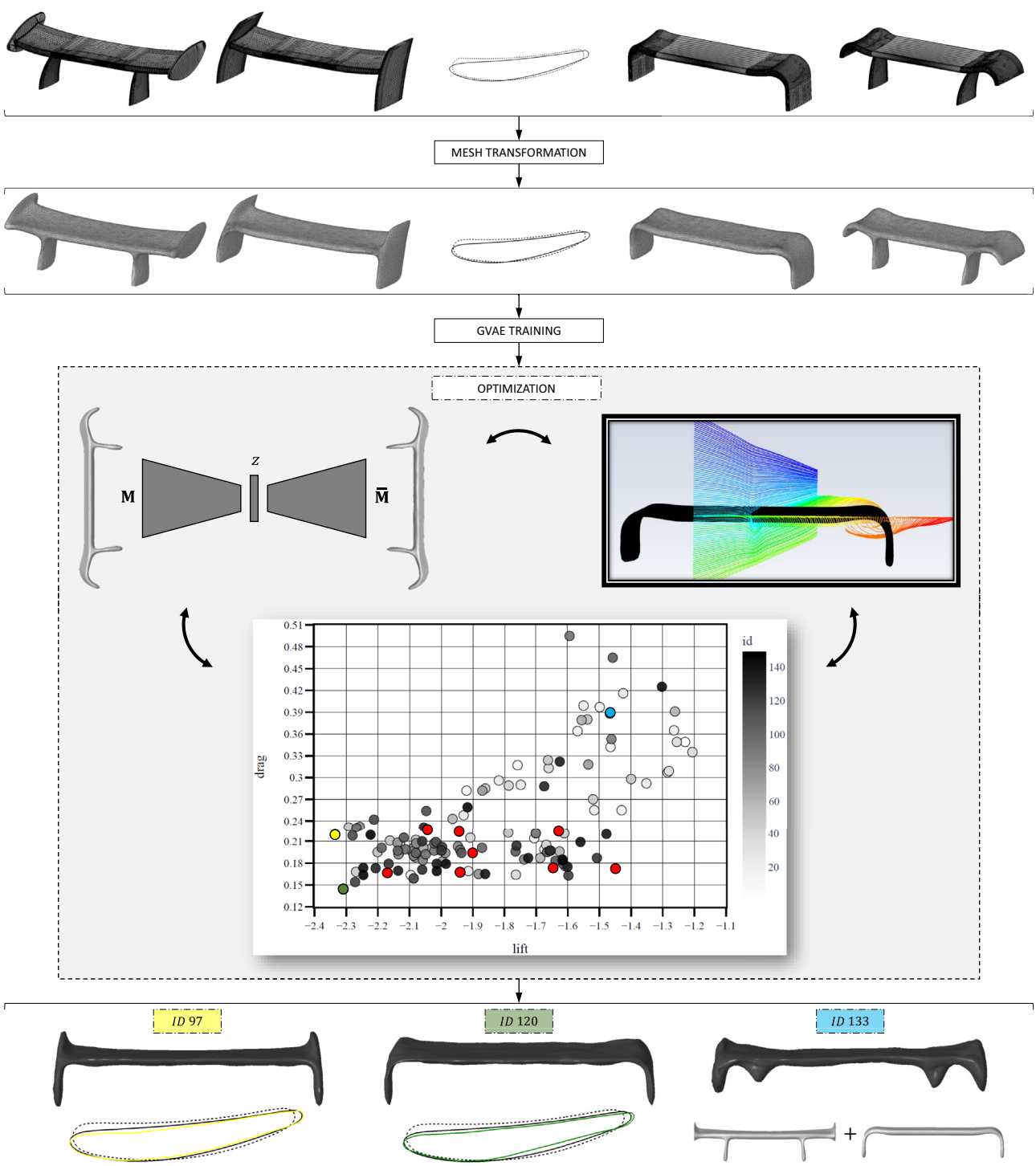

Fig. 6 ASO of the 3D cars' rear wing dataset. A total of eight different designs in *.stl format (four with a custom profile and another four with the NACA 6412) are first pre-processed to be later used by the GVAE network to learn the probabilistic latent space. After that, the optimization process starts, resulting in 150 new candidates depicted in the Pareto Front. In the end, two solutions are selected due to their outstanding aerodynamic features and their mid-plane profiles are compared with initial ones. At the same time, another geometry is selected due to its interpolation peculiarities

Table 2 Numerical values resulted from the optimation of the second dataset in comparison with the reconstructed parts. Lift and drag coefficients were obtained through ANSYS Fluent, while the latent vector is the encoding vector of each geometry in the GVAE architecture

		Drag	Lift	Latent vector			
Dataset	Custom model 1	0.222	− 1.614	− 0.9200	− 1.1114	0.2746	− 1.8901
	Custom model 2	0.193	− 1.898	− 1.5827	− 0.1155	− 1.3386	− 0.4524
	Custom model 3	0.161	− 1.939	− 1.2938	− 1.5788	− 1.1292	0.2964
	Custom model 4	0.177	− 1.444	0.0391	− 1.9142	− 0.2558	− 0.9271
	NACA model 1	0.225	− 1.927	− 0.6943	− 0.7930	− 0.2616	− 1.9384
	NACA model 2	0.231	− 2.061	− 1.4052	− 0.1674	− 1.5738	− 0.6106
	NACA model 3	0.157	− 2.173	− 0.9717	− 1.3922	− 1.2886	0.1854
	NACA model 4	0.167	− 1.646	− 0.0177	− 1.4567	− 0.4494	− 1.0061
Sim.	ID 97	0.220	− 2.335	− 1.3464	− 0.3337	− 1.7637	− 0.7826
	ID 120	0.144	− 2.309	− 1.0546	− 1.3556	− 1.7885	− 0.0115
	ID 133	0.390	− 1.478	− 1.3945	− 1.5479	− 1.8769	− 1.5201

discarded due to the presence of artifacts. This includes mesh reconstruction errors as a consequence of the lack of normal consistency or smoothness of the triangle elements and also mesh self-intersecting areas. Although 50 % of the detected failed meshes included less than 20 elements affected, since this initial geometry is the base to create a high-quality fluid control volume mesh (one of the most critical factors that must be considered to ensure simulation accuracy and convergence in CFD), all these candidates are automatically rejected. Their performance index is then associated with a bad value in order to avoid that the optimizer looks for another solution in that local minimum region. There is no doubt that this fact limits the exploration capability of the GA, however, from the resulting Pareto fronts can be observed that the final results are not excessively penalized. At this point, it is important to mention that, in case the number of discarded geometries increases, one way to ensure the convergence to the global optimal is to increase the population size.

- The resulting Pareto fronts (see Figs. 5 and 6) show that generated designs are well-distributed, which demonstrates the excellent dimensionality reduction properties of the proposed approach to conduct ASO. The method is successfully able to encode the non-linear properties of the input shapes and the minimal differences between them into a compact design space that is subsequently decoded. This allows the number of design variables to be drastically reduced from hundreds to a few while directly working with non-parametric 3D complex representations such as the *.stl files, which improves the convergence and computational cost without losing information. Furthermore, the generated fronts of non-dominated solutions have been able to fulfill all the empty gaps between the initial drag and lift coefficients. Therefore, once the process is finished, the user can decide which is the optimal design according to the application requirements. For example, going back to the airfoils' Pareto front, the user can select an airfoil that generates more lift with the associated increase in drag, another one with smaller lift and drag coefficients, or simply an intermediate solution.
- Visual inspection of the qualitative results depicted in Fig. 5 and 6 reveals the model is able to reconstruct realistic and effective geometries with intermediate properties of those of the training data. The origin of these intermediate drag and lift coefficients is the different shapes of the profiles, which leads to a distinct aerodynamic performance at profile level during the optimization process. Furthermore, in order to probe the large-scale interpolation properties in the second dataset, two pairs of two sufficiently different rear wings have been chosen to be encoded in their latent representations. Then, intermediate solutions have been produced by decoding the latent vector line that connects the samples of each pair as follows:

$$\mathbf{z} = a\mathbf{z}_1 + (1 - a)\mathbf{z}_2,$$

where a ∈ (0, 1). Figure 7 shows the interpolated mesh sequences produced that are successfully produced by the framework presented. In the same way, the method can interpolate newly generated shapes by interpolating their latent vectors to produce a deformation sequence.

8 Conclusion

This paper proposes a combination between a geometric deep learning architecture and a genetic algorithm to solve an aerodynamic shape optimization problem. It handles non-parametric 3D geometries in the form of triangle meshes, including datasets with file formats such as *.stl, *.ply, and *.obj. On the one hand, the machine learning framework is composed of a pre-processing frame in charge of normalizing and homogenizing all meshes to make them suitable

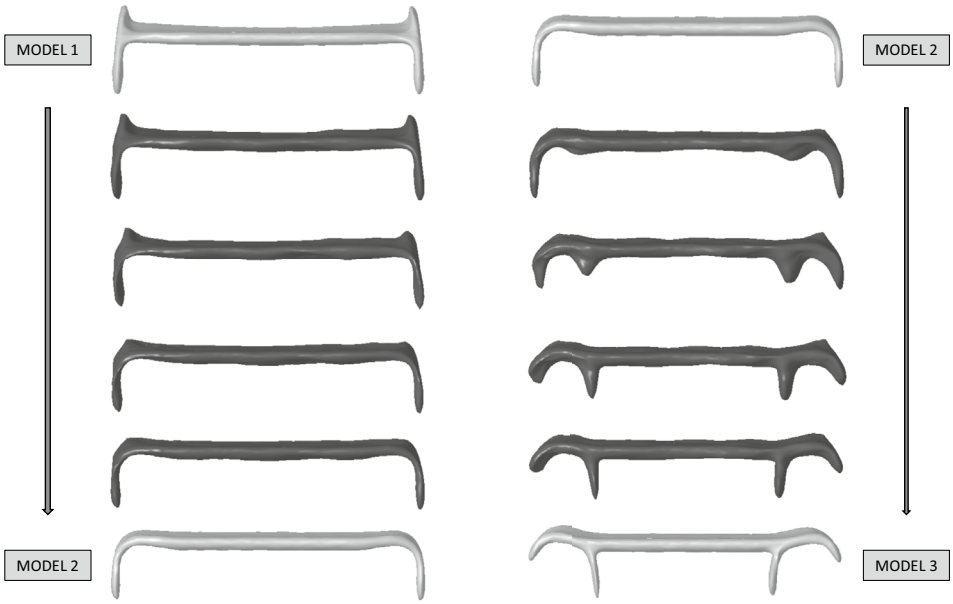

Fig. 7 Shape generation through linear interpolation between latent vectors of four different models that belong to the car's rear wings dataset. In each case, four intermediate solutions are generated by the trained probabilistic decoder

to be processed as graphs by the GVAE generative model. This neural network employs spline-based and Chebyshev spectral graph convolutional operators in order to capture the geometrical differentiating features between all the meshes of the dataset, being able to properly embed this shape variability in the latent space. On the other hand, the *NSGA-II* is the agent that guides the optimization process. With the objective of finding the optimal solution given a set of performance indices, it is in charge of generating new candidates by sampling the low-dimensional latent space and analyzing their aerodynamic potential through the connection with *ANSYS Fluent* for CFD calculations. Finally, a Pareto front with all available solutions is generated so as to decide which solution or set of solutions are the most suitable for a determinate application.

The results obtained after applying the framework to two different datasets show that the graph autoencoder is able to strongly encode the input dataset as a compressed representation in a reduced dimension, extract meaningful non-linear deformation patterns to create representations and perform 3D shape generation with great success. The experiments conducted demonstrated that geometries similar to the training data and solutions that get features from two different source meshes (interpolation) can be easily retrieved even if small datasets are employed. However, the ability to generate parts quite diverse from the ones of the training dataset (extrapolation) is still limited, which could be addressed by implementing post-processing steps that limit the appearance of artifacts, by exploring other generative networks like GANs applied to graphs or by using reasonably sized mesh datasets for supervised training (which undoubtedly might be difficult to capture or model). Combining geometries with different topologies is another open research available that currently needs the investment of too many resources until good-quality results can be achieved. The framework proposed is a shape optimization method and not a topological optimization one. Thus, it cannot deal with, for example, a dataset of wings with different numbers of flaps and spaces between them. Although these geometries are closed, they have a different number of holes and the grid connection is inherently diverse. This would require changing how parts are transformed into graphs or even using another non-Euclidean neural network architecture not directly based on graphs but on manifolds or even on surface reconstruction from projections.

Anyway, we have proved that the combination of this DL architecture together with the *NSGA-II* genetic algorithm has efficiently created geometries with mixed aerodynamic characteristics by fulfilling the empty gaps between those of the initial dataset; but it has also been able to generate new solutions with better performance indices. All of these allow the final users to perform ASO with non-parametric 3D files and to get as output a complete Pareto front from which they can select which is the better option for the required application. Moreover, it has been verified that treating the meshes as graphs and employing geometric deep learning operators is the correct way to handle detailed 3D data not only for this task but also for future applications. From now on, there is no need to rely on intermediate representations of 3D shapes like voxels or point clouds that suffers from high computational costs, low-resolution results, lack of smoothness, and limited inference.

Lastly, it is important to highlight that computation efficiency is a critical aspect in this kind of workflow in which

the decision-making is based on a heuristic algorithm and the performance is obtained through the connection with a CFD software. Another high-time-consuming process is related to the training of the network, especially when large datasets are employed. Nonetheless, since the bottleneck of the framework is clearly the CFD analysis, for future works the combination of the presented approach with reduced order model (ROE) architectures should be investigated to reduce such a large amount of resources and improve the implementation in industrial applications (Abadía-Heredia et al. 2022).

Acknowledgements The authors would like to thank Meta and Linux Foundation for creating, publicly sharing, and maintaining the open-source Pytorch and Pytorch Geometric frameworks for AI collaborative research. Furthermore, the authors also acknowledge the internal support of Nebrija University to carry out the PhD program in Industrial and Computer Science Technology (D019).

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Replication of results The results presented in this paper were obtained using a *Python* implementation of the proposed methodology and algorithms. The code to reproduce the case study is available through GitHub (https://github.com/JJab0n). Researchers or interested parties are welcome to contact the authors for further explanations.

References

Abadía-Heredia R, López-Martín M, Carro B, Arribas J, Pérez J, Le Clainche S (2022) A predictive hybrid reduced order model based on proper orthogonal decomposition combined with deep learning architectures. Expert Syst Appl 187(115):910. https://doi.org/10.1016/j.eswa.2021.115910

Achlioptas P, Diamanti O, Mitliagkas I, Guibas L (2018) Learning representations and generative models for 3D point clouds. In: Proceedings of the 35th international conference on machine learning, pp 40–49, https://doi.org/10.48550/ARXIV.1707.02392

Achour G, Sung W, Pinon O, Mavris D (2020) Development of a conditional generative adversarial network for airfoil shape optimization. In: AIAA SciTech Forum, https://doi.org/10.2514/6.2020-2261

Allen CB, Poole DJ, Rendall TC (2018) Wing aerodynamic optimization using efficient mathematically-extracted modal design variables. Optim Eng 19(2):453–477. https://doi.org/10.1007/s11081-018-9376-7

Asperti A, Trentin M (2020) Balancing reconstruction error and Kullback-Leibler divergence in variational autoencoders. IEEE Access 8:199440–199448. https://doi.org/10.1109/ACCESS.2020.3034828

Bellman R (1957) Dynamic programming. Princeton University Press, New Jersey

Ben-Hamu H, Maron H, Kezurer I, Avineri G, Lipman Y (2018) Multi-chart generative surface modeling. ACM Trans Graph 37(6):1–15. https://doi.org/10.1145/3272127.3275052

Berguin SH, Mavris DN (2014) Dimensionality reduction in aerodynamic design using principal component analysis with gradient information. In: 10th AIAA multidisciplinary design optimization conference, pp 71–87, https://doi.org/10.2514/6.2014-0112

Bishop CM (2006) Pattern recognition and machine learning. Springer, New York

Bouritsas G, Bokhnyak S, Ploumpis S, Bronstein M, Zafeiriou S (2019) Neural 3D Morphable models: spiral convolutional networks for 3D shape representation learning and generation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 7213–7222, https://doi.org/10.48550/ARXIV.1905.02876

Brock A, Lim T, Ritchie JM, Weston N (2016) Generative and discriminative voxel modeling with convolutional neural networks. arXiv https://doi.org/10.48550/ARXIV.1608.04236

Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P (2017) Geometric deep learning: going beyond Euclidean data. IEEE Signal Process Mag 34(4):18–42. https://doi.org/10.1109/MSP.2017.2693418

Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs. In: ICLR 2014 conference, https://doi.org/10.48550/ARXIV.1312.6203

Cai T, Luo S, Xu K, He D, Liu TY, Wang L (2020) GraphNorm: a principled approach to accelerating graph neural network training. arXiv https://doi.org/10.48550/ARXIV.2009.03294

Chen W, Fuge M (2018) BézierGAN: automatic generation of smooth curves from interpretable low-dimensional parameters. arXiv https://doi.org/10.48550/ARXIV.1808.08871

Chen W, Ramamurthy A (2021) Deep generative model for efficient 3d airfoil parameterization and generation. In: AIAA SciTech Forum, https://doi.org/10.2514/6.2021-1690

Chen W, Chiu K, Fuge M (2019) Aerodynamic design optimization and shape exploration using generative adversarial networks. In: AIAA SciTech Forum, https://doi.org/10.2514/6.2019-2351

Corbera S, Olazagoitia J, Lozano J (2016) Multi-objective global optimization of a butterfly valve using genetic algorithms. ISA Trans 63:401–412. https://doi.org/10.1016/j.isatra.2016.03.008

Deb K (2011) Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, New York

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197. https://doi.org/10.1109/4235.996017

Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: 30rd conference of NeurIPS, pp 3844–3852, https://doi.org/10.48550/ARXIV.1606.09375

Derksen R, Rogalsky T (2010) Bezier-PARSEC: an optimized aerofoil parameterization for design. Adv Eng Softw 41(7):923–930. https://doi.org/10.1016/j.advengsoft.2010.05.002

Fan H, Su H, Guibas L (2017) A point set generation network for 3D object reconstruction from a single image. in: ieee conference on computer vision and pattern recognition, pp 2463–2471, https://doi.org/10.1109/CVPR.2017.264

Fey M, Lenssen JE (2019) Fast graph representation learning with pytorch geometric. In: ICLR workshop on representation learning on graphs and manifolds, https://doi.org/10.48550/ARXIV.1903.02428

Fey M, Lenssen JE, Weichert F, Müller H (2017) SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In: IEEE/CVF conference on computer vision and pattern recognition, pp 869–877, https://doi.org/10.48550/ARXIV.1711.08920

Foster N, Dulikravich G (1997) Three-dimensional aerodynamic shape optimization using genetic and gradient search algorithms. J Spacecr Rocket 34:36–42. https://doi.org/10.2514/2.3189

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. Adv Neural Inform Process Syst. https://doi.org/10.48550/ARXIV.1406.2661

Grey ZJ, Constantine PG (2018) Active subspaces of airfoil shape parameterizations. AIAA J 56(5):2003–2017. https://doi.org/10.2514/1.J056054

Guo Y, Wang H, Hu Q, Liu H, Liu L, Bennamoun M (2020) Deep learning for 3D point clouds: a survey. IEEE Trans Pattern Anal Mach Intell 43(12):4338–364. https://doi.org/10.1109/TPAMI.2020.3005434

He X, Li J, Mader CA, Yildirim A, Martins JR (2019) Robust aerodynamic shape optimization: from a circle to an airfoil. Aerosp Sci Technol 87:48–61. https://doi.org/10.1016/j.ast.2019.01.051

Henaff M, Bruna J, LeCun Y (2015) Deep convolutional networks on graph-structured data. arXiv https://doi.org/10.48550/ARXIV.1506.05163

Hou CKJ, Behdinan K (2022) Dimensionality reduction in surrogate modeling: a review of combined methods. Data Sci Eng 7(4):402–427. https://doi.org/10.1007/s41019-022-00193-5

Jabón J, Corbera S, Barea R, Martín-Rabadán J (2023) An evolutive-deformation approach to enhance self-supporting areas in additive manufacturing designs. Comput Ind Eng 182(109):386. https://doi.org/10.1016/j.cie.2023.109386

Kenway GKW, Martins JRRA (2017) Buffet-onset constraint formulation for aerodynamic shape optimization. AIAA J 55(6):1930–1947. https://doi.org/10.2514/1.J055172

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. In: Proceedings of the 3rd ICLR, https://doi.org/10.48550/ARXIV.1412.6980

Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv https://doi.org/10.48550/ARXIV.1312.6114

Kolotouros N, Pavlakos G, Daniilidis K (2019) Convolutional mesh regression for single-image human shape reconstruction. In: IEEE/CVF conference on computer vision and pattern recognition, pp 4496–4505, https://doi.org/10.1109/CVPR.2019.00463

Kulfan B (2008) Universal parametric geometry representation method. J Aircr 45:142–1587. https://doi.org/10.2514/1.29958

Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Stat 22(1):79–86. https://doi.org/10.1214/aoms/1177729694

Lepine J, Guibault F, Trepanier JY, Pepin F (2001) Optimized nonuniform rational B-spline geometrical representation for aerodynamic design of wings. AIAA J 39(11):2033–2041. https://doi.org/10.2514/2.1206

Li J, Zhang M (2021) On deep-learning-based geometric filtering in aerodynamic shape optimization. Aerosp Sci Technol 112(106):603. https://doi.org/10.1016/j.ast.2021.106603

Li J, Zhang M, Martins J, Shu C (2020) Efficient aerodynamic shape optimization with deep-learning-based geometric filtering. AIAA J 58:1–17. https://doi.org/10.2514/1.J059254

Li J, Du X, Martins JR (2022) Machine learning in aerodynamic shape optimization. Prog Aerosp Sci 134(100):849. https://doi.org/10.1016/j.paerosci.2022.100849

Litany O, Bronstein A, Bronstein M, Makadia A (2018) Deformable shape completion with graph convolutional autoencoders. In: IEEE/CVF conference on computer vision and pattern recognition, pp 1886–1895, https://doi.org/10.1109/CVPR.2018.00202

Loshchilov I, Hutter F (2018) Decoupled weight decay regularization. In: ICLR 2019 conference on blind submission, https://doi.org/10.48550/ARXIV.1711.05101

Lyu P, Kenway G, Martins J (2015) Aerodynamic shape optimization investigations of the common research model wing benchmark. AIAA J 53(4):968–985. https://doi.org/10.2514/1.J053318

Masters DA, Taylor NJ, Rendall TCS, Allen CB, Poole DJ (2017) Geometric comparison of aerofoil shape parameterization methods. AIAA J 55(5):1575–1589. https://doi.org/10.2514/1.J054943

Maturana D, Scherer S (2015) VoxNet: a 3D convolutional neural network for real-time object recognition. In: IEEE/RSJ international conference on intelligent robots and systems, pp 922–928, https://doi.org/10.1109/IROS.2015.7353481

Moschoglou S, Ploumpis S, Nicolaou M, Papaioannou A, Zafeiriou S (2020) 3DFaceGAN: adversarial nets for 3D face representation, generation, and translation. Int J Comput Vision 128:2534–2551. https://doi.org/10.1007/s11263-020-01329-8

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. (2019) Pytorch: an imperative style, high-performance deep learning library. In: 33rd conference of NeurIPS, pp 8024–8035, https://doi.org/10.48550/ARXIV.1912.01703

Piegl L, Tiller W (1997) The NURBS book. Springer, New York

Poole DJ, Allen CB, Rendall TC (2019) Efficient Aero-structural wing optimization using compact aerofoil decomposition. In: AIAA Scitech Forum, https://doi.org/10.2514/6.2019-1701

Qi CR, Su H, Mo K, Guibas LJ (2017) PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 652–660, https://doi.org/10.48550/ARXIV.1612.00593

Qiu Y, Bai J, Liu N, Wang C (2018) Global aerodynamic design optimization based on data dimensionality reduction. Chin J Aeronaut 31(4):643–659. https://doi.org/10.1016/j.cja.2018.02.005

Rajnarayan DG, Ning A, Mehr JA (2018) Universal airfoil parametrization using B-splines. In: AIAA multidisciplinary analysis and optimization conference, https://doi.org/10.2514/6.2018-3949

Ranjan A, Bolkart T, Sanyal S, Black MJ (2018) Generating 3D faces using convolutional mesh autoencoders. In: Computer vision EECV, pp 725–741, https://doi.org/10.1007/978-3-030-01219-9_43

Ravi N, Reizenstein J, Novotny D, Gordon T, Lo WY, Johnson J, Gkioxari G (2020) Accelerating 3D deep learning with PyTorch3D. In: NeurIPS WiML workshop, https://doi.org/10.48550/ARXIV.2007.08501

Rogalsky T, Derksen RW, Kocabiyik S (1999) Differential evolution in aerodynamic optimization. Can Aeronaut Space J 46(4):183–190

Secanell M, Gamboa P, Suleman A (2006) Design of a morphing airfoil using aerodynamic shape optimization. AIAA J 44:1550–1562. https://doi.org/10.2514/1.18109

Sigmund O, Maute K (2013) Topology optimization approaches. Struct Multidisc Optim 48:1031–1055. https://doi.org/10.1007/s00158-013-0978-6

Sobieczky H (1999) parametric airfoils and wings. In: Notes on numerical fluid mechanics, pp 71–87, https://doi.org/10.1007/978-3-322-89952-1_4

Tan Q, Gao L, Lai YK, Xia S (2018a) Variational autoencoders for deforming 3D mesh models. In: IEEE/CVF conference on computer vision and pattern recognition, pp 5841–5850, https://doi.org/10.1109/CVPR.2018.00612

Tan Q, Gao L, Lai YK, Yang J, Xia S (2018b) Mesh-based autoencoders for localized deformation component analysis. Proc AAAI Conf Artif Intell 32(1):2452–2459. https://doi.org/10.1609/aaai.v32i1.11870

Tretschk E, Tewari A, Zollhofer M, Golyanik V, Theobalt C (2020) DEMEA: deep mesh autoencoders for non-rigidly deforming objects. In: Computer vision EECV, pp 601–617, https://doi.org/10.1007/978-3-030-58548-8_35

Viswanath A, Forrester AIJ, Keane AJ (2011) Dimension reduction for aerodynamic design optimization. AIAA J 49(6):1256–1266. https://doi.org/10.2514/1.J050717

Viswanath A, Forrester AIJ, Keane AJ (2014) Constrained design optimization using generative topographic mapping. Chin J Aeronaut 52(5):1010–1023. https://doi.org/10.2514/1.J052414

Wang N, Zhang Y, Li Z, Fu Y, Liu W, Jiang YG (2018) Pixel2Mesh: generating 3D mesh models from single RGB images. In: Computer vision EECV, pp 55–71, https://doi.org/10.1007/978-3-030-01252-6_4

Wang W, Ceylan D, Mech R, Neumann U (2019) 3DN: 3D deformation network. In: IEEE/CVF conference on computer vision

and pattern recognition, pp 1038–1046, https://doi.org/10.1109/CVPR.2019.00113

Wu J, Zhang C, Xue T, Freeman B, Tenenbaum J (2016) Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. Adv Neural Inform Process Syst. https://doi.org/10.48550/ARXIV.1610.07584

Yan X, Zhu J, Kuang M, Wang X (2019) Aerodynamic shape optimization using a novel optimizer based on machine learning techniques. Aerosp Sci Technol 86:826–835. https://doi.org/10.1016/j.ast.2019.02.003

Yuan YJ, Lai YK, Yang J, Duan Q, Fu H, Gao L (2020) Mesh variational autoencoders with edge contraction pooling. In: IEEE/CVF Conference on computer vision and pattern recognition workshops, pp 1105–1112, https://doi.org/10.1109/CVPRW50498.2020.00145

Zhang A, Lipton ZC, Li M, Smola AJ (2021) Dive into deep learning. arXiv https://doi.org/10.48550/ARXIV.2106.11342

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.