Towards Universal Parameterization: Using Variational Autoencoders to Parameterize Airfoils

Swannet, K.; Varriale, Carmine; Doan, Nguyen Anh Khoa

DOI
10.2514/6.2024-0686

Publication date
2024

Document Version
Final published version

Published in
Proceedings of the AIAA SCITECH 2024 Forum

AIAA SciTech Forum
8-12 January 2024, Orlando, FL
AIAA SCITECH 2024 Forum

10.2514/6.2024-0686

Towards Universal Parameterization: Using Variational Autoencoders to Parameterize Airfoils

Kilian Swannet*, Carmine Varriale†, Nguyen Anh Khoa Doan‡
Delft University of Technology, Delft, 2629 HS, The Netherlands

A design can only be as good as its mathematical representation. In engineering design optimization, the chosen method of parameterization can have significant impact on the outcomes. This paper introduces a novel methodology for airfoil design parameterization utilizing variational autoencoders (VAEs), a class of neural networks known for their proficiency in reducing dimensionality. However, a significant challenge with VAEs is the interpretability of the encoded latent space. This work aims to address this issue by creating a network with an interpretable latent space, yielding parameters that are understandable to humans. The effectiveness of this approach is evaluated using the comprehensive UIUC airfoil database, which offers a diverse range of airfoil shapes for analysis. We show that a VAE can successfully extract key features of airfoil geometries and parameterize them using six parameters, which show a clear correlation with airfoil properties in a way that remains understandable by the designer. Additionally, it smoothly interpolates the data points, allowing the generation of new airfoils and thus offering a practical and interpretable airfoil parameterization.

I. Nomenclature

$\vec{\cdot}$	=	Vector variable
$\tilde{\cdot}$	=	Normalized variable
x	=	Input
x'	=	Reconstructed input or generated output
Z	=	Latent parameter
n	=	Number of latent parameters
μ	=	Mean
σ	=	Standard deviation
$\mathcal{N}(\mu, \sigma)$	=	Gaussian distribution
$\mathcal{U}(a, b)$	=	Uniform distribution in range $[a, b]$
ε	=	Random value
β	=	Weight factor for KL divergence
$i,\ j$	=	Latent parameter index
L_{train}	=	Training loss
L_{recon}	=	Reconstruction loss
D_{KL}	=	Kullback–Leibler divergence
$\mathcal{P}$	=	Airfoil property values
δ	=	Value range
w	=	Correlation coefficient weights
t_{max}	=	Maximum thickness
$x_{t_{\text{max}}}$	=	Chordwise max thickness location
c_{max}	=	Maximum camber
$x_{c_{\text{max}}}$	=	Chordwise max camber location
R_{LE}	=	Leading edge radius
θ_{LE}	=	Leading edge angle
θ_{TE}	=	Trailing edge angle
γ_{TE}	=	Trailing edge wedge angle

II. Introduction

The rise of neural network technologies in various domains, including aerospace engineering, has introduced new methods and opportunities for engineering design. Neural networks are particularly valued for their universal function approximation capabilities [1], which can be used to uncover hidden patterns within datasets. However, a major issue with these networks is their lack of interpretability, due to the complexity of their internal processes. This problem, often referred to as the "black box" issue, limits their use in practical design scenarios where understanding the influence of design parameters on design performance is crucial.

This study aims to leverage the function approximation capabilities of neural networks to assess the potential of generative networks in providing a universally applicable, automated approach to parameterization. This involves

*PhD Candidate, Flight Performance and Propulsion Section, Faculty of Aerospace Engineering, k.swannet@tudelft.nl
†Assistant Professor, Flight Performance and Propulsion Section, Faculty of Aerospace Engineering, c.varriale@tudelft.nl; AIAA Member
‡Assistant Professor, Aerodynamics Section, Faculty of Aerospace Engineering, n.a.k.doan@tudelft.nl

reducing data dimensionality, finding a function that can represent this data with as few parameters as possible, while simultaneously procuring a function which can reproduce the original data from the low-dimensional representation.

The focus of this work specifically is on exploring how interpretable the latent space of variational autoencoders (VAEs) can be when applied to airfoil design parameterization. The goal is to find an approximation of a parameterization function for airfoil shapes that provides a low-dimensional, interpretable representation. While interpretability is not always necessary for applications like optimization problems, we want to make these models more transparent and user-friendly for engineers. By establishing clear links between design parameters, we strive to make these advanced models not just efficient, but also practically useful in the field of aerospace engineering.

III. Previous work

Generative algorithms, notably Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), have gained substantial attention in recent research [2–4] for aerodynamic surrogate modeling and flow prediction [5, 6] and generation of new data points to augment sparse datasets [7–9]. These methods play a pivotal role in expanding the capabilities of computational design and data analysis.

GANs are particularly noted for their ability to augment limited datasets, thereby improving their utility in training surrogate models [7]. Despite their effectiveness, GANs often exhibit challenges in terms of output control and input interpretability, issues that are addressed in our study. VAEs, on the other hand, have been effectively employed in synthesizing airfoil geometries by creating a compact, low-dimensional latent representation of their shape for use in optimization tasks [10]. The work of Yonekura and Suzuki [11] further advanced this application by incorporating a conditional label into the VAE framework. This modification allows for the generation of airfoils based on both latent variables, which describe the shape, and an additional label that specifies desired aerodynamic characteristics. Yang et al. [12] took this one step further and trained an autoencoder on three dimensional geometries, enabling the generation of novel shapes that merge various objects. This showcases the VAE's capacity for creating innovative and blended designs for conceptual design.

While VAEs and GANs have both seen significant progress in their applications, there remains a notable lack of interpretability of their latent spaces, particularly for engineering design parameterization [13]. Interpretability is essential as it fosters trust in the model by enabling engineers to understand and intuitively interact with it. This clarity is also vital for diagnosing and correcting errors, making accurate design choices, and seamlessly integrating human expertise with automated processes. While the internal process of a neural network remains inexplicable, an understandable latent space addresses, at least partially, the "black box" problem.

The current work primarily focuses on VAEs, given their favorable training stability compared to GANs [14], and their inherent ability to define the latent distribution of features more explicitly [15]. Moreover, the adaptability of VAEs is enhanced by the ability to modulate latent space disentanglement through adjustments in the weight parameter within their loss function [13, 16]. Disentanglement refers to each latent variable capturing distinct properties of the data. This results in a representation where changes in one dimension correspond to changes in one feature of the data, enhancing interpretability and ease of manipulation. This adaptability positions VAEs as a more suitable choice for tackling the challenges of interpretable parameterization and generation in this study.

IV. Generative Machine Learning Models

Generative modeling involves creating a low-dimensional, probabilistic representation of data, enabling the generation of new data points through sampling [17]. This probabilistic representation is commonly referred to as the latent representation. In the context of our study, the latent representation serves as the core feature of interest in the design and analysis of airfoils. Based on the reviewed literature, the VAE network architecture has been identified as the most suitable method for inferring a disentangled latent space from a dataset of airfoil geometries. The VAE's ability to efficiently encode data into a compact latent space and then decode it back to its original form makes it particularly well-suited for this task. This section begins with an introduction to the concept of an autoencoder in subsection IV.A, followed by a detailed presentation of the VAE in subsection IV.B, including an explanation of how the latent space of the VAE can be regularized to promote disentanglement and enhance interpretability.

A. Autoencoder Networks

An autoencoder (AE) is a type of neural network architecture designed to replicate its input data through a process of feature extraction and data generation [18]. The architecture of an AE is composed of two main segments, the encoder

and the decoder, as illustrated in Figure 1. The encoder network is responsible for compressing the input data vector, denoted as $\bar{x}$, into a condensed form of latent variables or features, represented by the latent vector $\bar{Z}$. The bar notation signifies that these are vectors, which are specifically rank-one tensors in the context of machine learning algorithms. This process involves extracting the essential characteristics of the input data, capturing the underlying patterns and relationships into a more compact and efficient representation. The primary function of the decoder is to reconstruct the original input data using the latent features provided by the encoder. The output vector of the decoder, denoted as $\bar{x}'$, is an approximation of the original input. This reconstruction process tests the ability of the latent representation to retain the significant features of the input data, ensuring that critical information is not lost during compression.

The training of AEs comprises reconstruction of many samples, and minimizing the difference between the original input data and its reconstructed version, known as the reconstruction error. Effective training enables the encoder and decoder to function accurately and independently. Post-training, the encoder serves as a tool for dimensionality reduction, particularly useful in simplifying complex problems and facilitating comparisons across diverse datasets [19]. The decoder can now function as a deterministic generator, capable of constructing an output based on specific latent variables. This functionality lends itself well to be used for data compression, and shows potential for design parameterization. One of the key advantages of AEs is their ability to perform a more advanced, nonlinear form of Principal Component Analysis (PCA). This capability allows AEs to serve as a robust alternative to the traditional PCA eigenvector approach, often outperforming it in terms of capturing complex data patterns and relationships [18, 20, 21]. The performance and versatility of AEs can be augmented by incorporating regularization techniques, which involve adding specific terms to the training loss function. These terms introduce additional constraints or goals, extending the autoencoder's utility beyond mere data reconstruction. For example, a term can be included to encourage sparsity in the latent representation, leading to more distinct and less redundant feature encoding, or a term to minimize the derivatives of the representation, enhancing its smoothness and therefore a smoother transition between generated outputs. Additionally, a loss term can be tailored to subtly alter the generated outputs in comparison to the inputs, such as fine-tuning the wing planform or airfoil shape in a manner that improves aerodynamic properties like drag [11, 18].

B. Variational Autoencoders

The Variational Autoencoder (VAE), first presented by Kingma et al. [16], is a variant of the AE. It incorporates a probabilistic layer at the end of the encoder, transforming the deterministic framework of a conventional AE into a probabilistic one. This novel architecture, shown in Figure 1 accounts for the inherent uncertainty and variability present in real-world data. The main functionality of the encoding portion of a VAE is characterized by its ability to

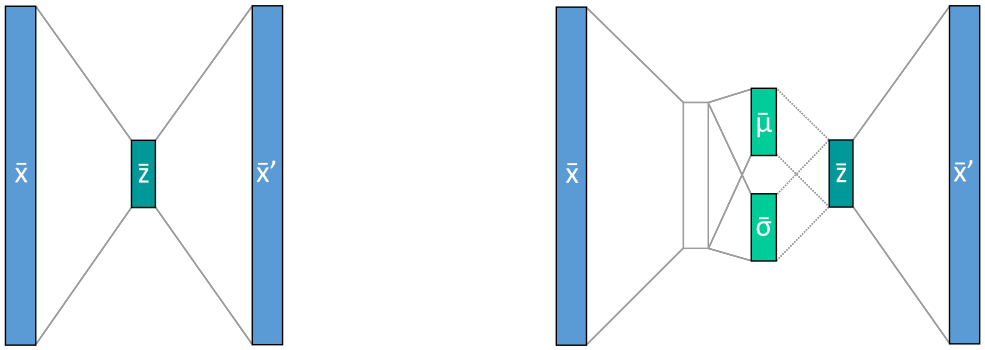

Fig. 1 The autoencoder (left) and variational autoencoder (right) network architectures.

capture the posterior probability distribution (defined by the mean μ, and standard deviation σ) of the latent variables or features, conditional on the input data. Mathematically, this can be represented as $p_{\text{enc}}(\bar{Z}|\bar{x})$. The ability to ascertain this distribution is vital, as it enables the network to infer the likely values of the underlying latent variables given the observed input data. The decoding portion, on the contrary, seeks to understand how the latent variables or features might influence or generate the observed data, by modeling the likelihood distribution of the known data given the latent variables. This is typically denoted as $p_{\text{dec}}(\bar{x}|\bar{Z})$ [18]. Just like the standard AE, the encoder and decoder collaborate to accomplish two integral tasks in the operation of a VAE. The first task is the encoding of input data into the latent space, effectively parameterizing the input. This involves inferring the latent distributions of the data, where the encoder translates the input data into a set of probabilistic distributions in the latent space, as outlined in Equation 1. These latent

distributions are characterized by parameters such as mean and variance or standard deviation. This step is distinct from traditional AEs, where the encoding is deterministic. Subsequently, the values of the latent parameters are sampled using the reparameterization trick, described in Equation 2 [18], where $\odot$ indicates elementwise multiplication. This process involves taking the mean of the latent distribution and adding to it the standard deviation multiplied by a random value ε sampled from a standard Gaussian distribution $\mathcal{N}(0, 1)$.

$$\bar{\sigma}, \bar{\mu} = f_{\text{enc}}(\bar{x}) \tag{1}$$

$$\bar{z} = \bar{\mu} + \bar{\varepsilon} \odot \bar{\sigma}, \quad \text{where } \bar{\varepsilon} \sim \mathcal{N}(0, 1) \tag{2}$$

The second key task of a VAE is the generation of new data instances by providing the sampled latent vector as input to the decoder. The decoder now functions as a generator and outputs the reconstructed (or newly generated) data, as defined by Equation 3. The inherent variability in the probabilistic representation of the latent space facilitates the creation of a wide range of plausible outputs. Each sampled point from the latent space represents a unique combination of features, which the decoder then translates back into a data point. This capability allows the VAE to generate new data instances that differ from those in the original training set, demonstrating its utility in applications requiring the generation of novel, yet plausible, data variations.

$$\bar{x}' = f_{\text{dec}}(\bar{Z}) \tag{3}$$

Training a Variational Autoencoder (VAE) involves a delicate balance between reconstructing the input data and shaping the latent space to adhere to a specific probabilistic distribution. The reconstruction loss measures the difference between the original input, and the reconstruction returned by the decoder, measuring how much information is lost passing through the network. The Kullback–Leibler (KL) divergence ($\mathrm{D_{KL}}$), obtained using Equation 4, plays a central role in this process. Integrated into the loss function, Equation 5, it represents the difference between the inferred latent distribution and a Gaussian distribution. Therefore, it pushes the latent parameter distributions towards a standard Gaussian distribution [16]. The weight factor β is pivotal in determining the relative importance of the KL divergence compared to the reconstruction loss. This process is referred to as regularization [18].

$$\mathrm{D_{KL}} = -\frac{1}{2} \sum \left(1 + \log(\bar{\sigma}^2) - \bar{\mu}^2 - \bar{\sigma}^2\right) \tag{4}$$

$$\mathrm{L_{train}} = \mathrm{L_{recon}} + \beta \cdot \mathrm{D_{KL}} \tag{5}$$

When β is assigned a low value or set to zero, the training process prioritizes minimizing reconstruction loss, leading the VAE to function similarly to a standard autoencoder (AE). However, this approach can result in overfitting of the latent parameters, leading to poor generalization and less effective interpolation between data points. In such cases, the model may struggle to accurately reconstruct or generate new data points that are not closely aligned with the training set.

Assigning a higher value to β enhances the impact of the KL divergence in the loss function, leading to a smoothing effect on the learned features. This adjustment typically results in worse performance in pure data reconstruction, as the model may not capture every feature of every data point. Nevertheless, it promotes better generalization and smoother interpolation between learned data points. This can be particularly beneficial in applications where the generation of new, plausible data points is more important than the precise replication of existing ones. The KL divergence encourages the latent parameter distributions to assume the shape of a standard Gaussian. As a result, redundant latent variables will simply resemble Gaussian noise, which is filtered out by the decoder network and will thus have no effect on the generated output. This selective focus could lead to the disentanglement of the latent space, resulting in a more interpretable and manageable representation. While this also means that fewer details of the data are retained, it allows the latent space to concentrate on the more salient features of the data. As a result, the model utilizes only as many latent parameters as necessary to represent the bulk of the data.

However, if β is set too high, the emphasis shifts excessively towards the KL divergence, overshadowing the reconstruction loss. In such scenarios, the generator network may fail to reproduce the variety present in the training data, instead producing outputs that converge towards a generalized average of the data, regardless of the latent parameter values. This situation highlights the need for a careful selection of β to achieve an optimal balance. What constitutes a low or high value depends on the problem at hand. The goal is to create a network capable of capturing the key features of data in as few dimensions as possible, finely balancing the detailed reproduction of training data with the ability to generalize and interpolate between data points, thereby generating coherent and smooth shapes.

V. Methodology

The development of the VAE network for this study is implemented using Pytorch [22], with its detailed architecture and configuration presented in subsection V.C. First, however, subsection V.A will discuss the training data used, and how different datasets are set up, and subsection V.B specifies the calculation of the reconstruction loss. Finally, subsection V.D will describe the process of choosing the value for the weighing factor β for the KL divergence in the loss function.

A. Data Setup

The network is trained on a dataset containing 1619 airfoil data points, derived from the UIUC airfoil database [23], a comprehensive online resource that provides geometric data for a wide range of airfoil shapes. To standardize the data, all airfoil coordinate files are interpolated, resulting in surface coordinates at identical cosine-spaced chordwise locations. Each airfoil is represented by 199 points, with 100 points each for the upper and lower surfaces, excluding the duplicated leading edge point. The trailing edge points are retained on both surfaces to accommodate airfoils that exhibit non-zero trailing edge gaps or thicknesses. Given the constant chordwise locations, the network uses only the y ordinates of each point as input, simplifying the input data structure.

To facilitate the training and evaluation of the network, this dataset is split into three subsets: 75% of the data is used for training, 15% for validation, and the remaining 10% is allocated for testing purposes. The training set is primarily used for backpropagation, a fundamental mechanism in neural network training. Backpropagation is an iterative process that involves two key phases: the forward pass and the backward pass. In the forward pass, input data is fed through the entire network, resulting in an output. This output is then compared to the desired outcome, and the difference between them is quantified as the reconstruction loss (L_{recon}). The backward pass is where backpropagation plays a critical role. In this phase, the loss is propagated back through the network in the reverse direction. This process involves the computation of gradients for each parameter (weights and biases) of the network, using algorithms like gradient descent. The gradients indicate how much each parameter should be adjusted to minimize the loss. By iteratively adjusting these parameters, the network gradually learns to reduce the error in its predictions. The validation set, on the other hand, is utilized to monitor the network's performance. Specifically, it helps track the average loss per training iteration, called an epoch, providing insights into how well the training is progressing. This set is crucial for ensuring that the model is learning effectively and not just memorizing the training data, a phenomenon known as overfitting. Lastly, the test set plays a vital role in hyperparameter tuning, a process elaborated in subsection V.C. Hyperparameters are the settings or configurations that govern the overall behavior of the neural network, such as the learning rate, number of layers, and size of each layer. The validation set is used to evaluate different hyperparameter configurations, ensuring that the chosen settings optimize the network's performance on unseen data. This is crucial for developing a model that not only performs well on the training data but also generalizes effectively to new, unseen data.

B. Loss Calculation

The loss function and the KL divergence were previously defined in Equation 5 and Equation 4 respectively. For the current study, the reconstruction loss is computed using the relative mean squared error, as specified in Equation 6. This method quantifies the difference between the original input data and the data reconstructed by the network, relative to the value of the original. To ensure stability in the calculation and avoid division by zero, a small modification is made to the y ordinates: they are all offset upwards by one, a technique indicated by the +1 subscript in the equation. Relative mean squared error was chosen as it slightly emphasizes lower values. Since the airfoil coordinates use cosine spacing, this will help shape the leading and trailing edges, where points will be concentrated and small irregularities will have a larger effect on the airfoil surface smoothness.

$$L_{recon} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\bar{x}_{+1} - \bar{x}'_{+1}}{\bar{x}_{+1}}\right)^2 \tag{6}$$

C. Hyperparameter Selection

Determining the optimal hyperparameters for both the network architecture and training process in a VAE model generally still relies on heuristic methods, involving a mix of empirical evidence, theoretical understanding, and often a bit of experimentation, as there is still a significant element of experience to this science. For the training hyperparameters, a balanced approach was chosen. The batch size is the number of training samples processed before

the model's internal parameters are updated by the optimizer. It is known that fewer, larger batches during training can lead to faster convergence, but at the risk of poorer generalization, whereas a higher number of smaller batches typically promotes better generalization at the cost of speed [18, 24]. In light of this, the batch size was set to 64, aiming to strike a balance between convergence speed and generalization capability. The learning rate determines the size of the steps the algorithm takes during optimization, and was chosen as 3×10^{-4}, following recommendations from best practices in the field [25].

Rather than defining a maximum number of epochs, a more adaptive approach for determining the duration of training was utilized. The training process is designed to halt when the training loss does not improve over 250 epochs. With this form of early stopping a balance is struck between sufficient training to capture the complexities of the data and avoiding excessive training that could lead to overfitting.

To estimate the number of required latent parameters, a PCA was conducted on the dataset. Observing the variance ratio of the first ten components, as shown in a Scree plot in Figure 2, revealed that the majority of data variance is captured by the first three components, with additional details captured by an additional two components. This suggests that most of the critical information present in the data can be encapsulated in three parameters. However, to ensure robustness and to capture more subtle features in the data, the network was designed with eight latent parameters. This decision is based on the assumption that the VAE should be able to match or exceed the dimensionality reduction achieved by PCA. By choosing eight latent parameters, the network is given enough flexibility to learn and represent the essential features of the airfoils, while trusting that the KL divergence loss term will render redundant parameters inactive.

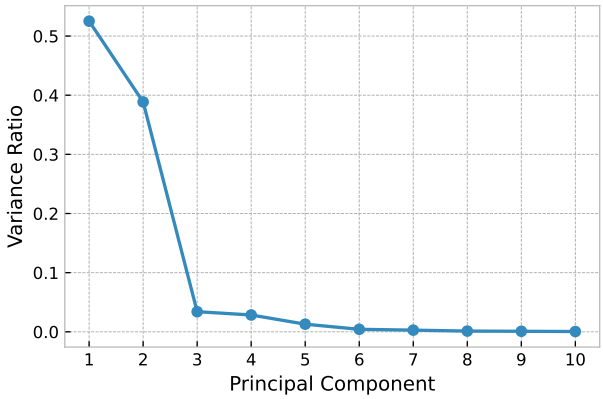

Fig. 2 Scree plot showing the variance ratio of the first ten principal components of the training data.

The network itself is composed solely of linear layers with Leaky ReLU activation functions. This leaky version of a rectified linear unit function, provided in Equation 7, is preferred over the standard ReLU function as it allows negative values to retain significance in the network, though with a reduced value. The final layer of the decoder utilizes a hyperbolic tangent activation function to ensure that the output ordinates fall within the range $y_i \in [-1, 1]$.

$$f(x) = \max(0.1x,\ x) \tag{7}$$

To determine the number of layers in the network, and the number of neurons in each layer, a hyperparameter optimization was conducted using Optuna [26]. This optimization process was configured to maintain the characteristic hourglass shape of the autoencoder, ensuring that each successive layer had an equal or lower number of neurons than the preceding one, and allowed a maximum of five hidden layers. The encoder and decoder use an equal but mirrored network layout. The Optuna study ran for 1000 trials, employing a tree-structured parzen estimator sampler with a hyperband pruner. The testing dataset served as a benchmark for evaluating the performance of each network architecture proposed by the Optuna optimizer, ensuring that the network was tested on data it hadn't encountered during training.

The architecture deemed optimal by Optuna, as illustrated in Figure 3, consisted of four hidden layers with neuron counts of 196, 172, 172, and 143, respectively. This structure was mirrored in both the encoder and decoder networks. Notably, repeated optimization studies with Optuna yielded varying results, with no consistent pattern emerging in the proposed optimal network sizes. Intriguingly, a simpler network configuration with significantly fewer neurons (comprising four layers with 128, 64, 32, and 16 neurons, respectively) demonstrated performance nearly identical to the

Optuna-optimized network. These findings hint that the effectiveness of the network in capturing data features does not linearly correlate with an increase in complexity beyond a certain threshold, and that as long as the network has a sufficient number of nodes to approximate a function which captures the key features of the data, adding more nodes beyond a certain point provided diminishing returns.

Despite these observations, the network architecture identified in the initial Optuna study was selected for implementation in this project. This decision is grounded in the confidence derived from its construction within the established and tested library that is the Optuna framework.

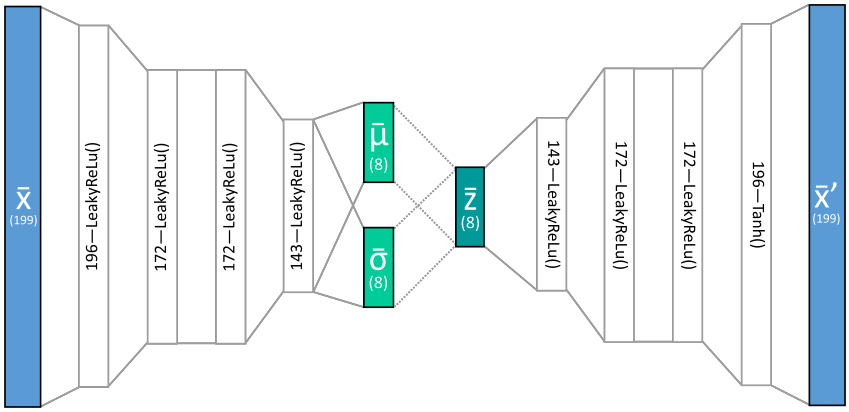

Fig. 3 Final network architecture used to obtain the results discussed in this paper.

D. Determining β

The parameter β plays a pivotal role in controlling the level of disentanglement in the latent space, which directly impacts the interpretability and dimensionality reduction capabilities of the model. However, setting the value of β presents a unique challenge, as it cannot be effectively included in the hyperparameter optimization process. This is because the inclusion of the KL divergence term in the loss function, which β influences, inherently increases the reconstruction and total loss. An optimization algorithm would likely set β to zero to minimize the total loss, thus negating its intended effect.

To address this issue, Kou et al. [10] employed the Fréchet Distance between the distributions of the airfoil coordinates of the original and reconstructed dataset. This metric was used to establish a lower bound for the value of β. They then determined the optimal value of β by minimizing both the Fréchet distance and the reconstruction loss. Notably, the appropriate value of β coincided approximately with the point where the reconstruction loss began to increase from its minimum observed value.

The average reconstruction loss, total loss, and scaled KL divergence were plotted across a range of β-values, as shown in Figure 4. Upon evaluating the models that contributed to this plot, the value of $\beta = 4.83 \times 10^{-7}$ was chosen for the current network training. The respective model demonstrates satisfactory performance, generating relatively smooth airfoils and capturing distinct features in the latent space. The chosen value of β is positioned just before the point where the importance of the reconstruction loss and scaled KL divergence begin to equalize. This strategic placement ensures that while the model prioritizes accurate reconstruction, it also effectively leverages the benefits of the KL divergence term for a balanced and efficient representation in the latent space.

VI. Validation

Post-training, the encoder and decoder networks of the VAE serve distinct but complementary functions. The encoder is now capable of determining the latent parameterization of airfoils, effectively encoding the essential features of each airfoil into a compact latent space. Conversely, the decoder functions as a generator, capable of producing airfoils from specified sets of latent parameters.

A critical step following training is to assess the performance of the VAE in reproducing the airfoils it was trained on. This evaluation involves passing all airfoils from the dataset through the entire network and comparing the reconstructed airfoils with their original counterparts. The reconstruction validation for the airfoils that exhibited the

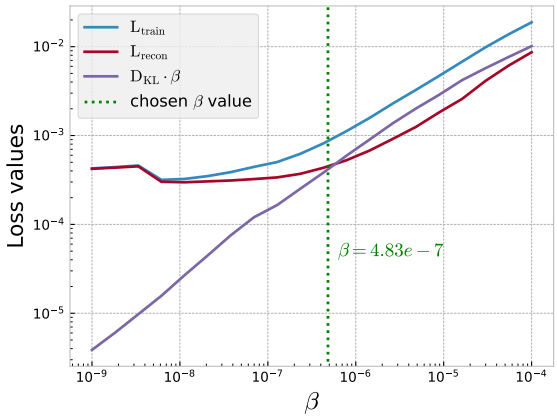

Fig. 4 Evolution of average reconstruction loss, average total loss, and average scaled KL-Div with increasing β.

highest reconstruction loss is illustrated in Figure 5, focusing on the 12 airfoils with the poorest reconstruction. The respective reconstruction loss of each airfoil is indicated in each plot.

Visual analysis of these airfoils hints at specific challenges and insights:

- AH93-W-480B: This airfoil was not accurately reconstructed, potentially due to its large trailing edge gap, a feature that might be underrepresented in the dataset. This suggests that the network struggled to handle shapes with such a distinctive characteristic.
- UA79-SF-187: The reconstruction issue here appears to be related to an improper rotation, as evidenced by the trailing edge not aligning with the $(x, y) = (1, 0)$ coordinate. This could be due to inconsistencies in the original airfoil coordinate files, such as missing or mislocated leading edge points, affecting the alignment process.
- HAM-STD-HS1-430: Discrepancies near the trailing edge are observed, likely for reasons similar to the AH93-W-480B, where the network's representation of airfoils with larger trailing edge gaps is less accurate, and skewed towards a trailing edge shape observed for the FX79-W-660A, which was mostly correctly reconstructed.

Despite these specific instances of inadequate reconstruction, the overall performance of the autoencoder is commendable, especially given the roughness of some entries in the dataset it was trained on. Notably, the network shows an ability to correct likely errors in the dataset, such as the overly sharp leading edge of the GOE440 and the misplaced point on the bottom surface of the StCYR171. The quality of the generated airfoils can only be as good as the ones used for training. Decisions regarding the handling of outlier airfoils, like those with large trailing edge gaps or errors in the coordinates, clearly impact the network's learning. While removing these outliers could potentially improve the overall performance, this would require manual verification of each airfoil in the dataset, a labor-intensive process, as well as reduce the available training data. Currently, work is underway on developing an airfoil geometry processing tool aimed at automating the processing of airfoils in the database, which should enhance the quality of the datapoints and, by extension, the performance of the VAE, without reducing the size of the dataset.

In conclusion, the reconstruction validation, particularly of the airfoils with the worst reconstruction loss demonstrates that overall the VAE performs satisfactorily, especially given the roughness and sparsity of some of the datapoints. The insights gained from this evaluation will inform future improvements and refinements in both the dataset and the VAE model.

VII. Results

By processing the entire dataset solely through the encoder network of the autoencoder, one can sample the latent parameters Z_i from the probability distributions inferred by the encoder. The resulting latent parameter distributions are plotted in Figure 6, revealing that all latent parameters closely resemble a standard Gaussian distribution. While this alignment with a Gaussian distribution is indicative of the effectiveness of the KL divergence term in the loss function, it does not provide direct insight into the disentanglement of the latent space. Determining how many and which specific features have been effectively extracted by the encoder from the data requires more in-depth investigation

To get an idea of the degree of disentanglement of the latent space, the effects of each latent parameter on the produced airfoil are investigated. Airfoils are generated by systematically setting the value of each latent parameter two

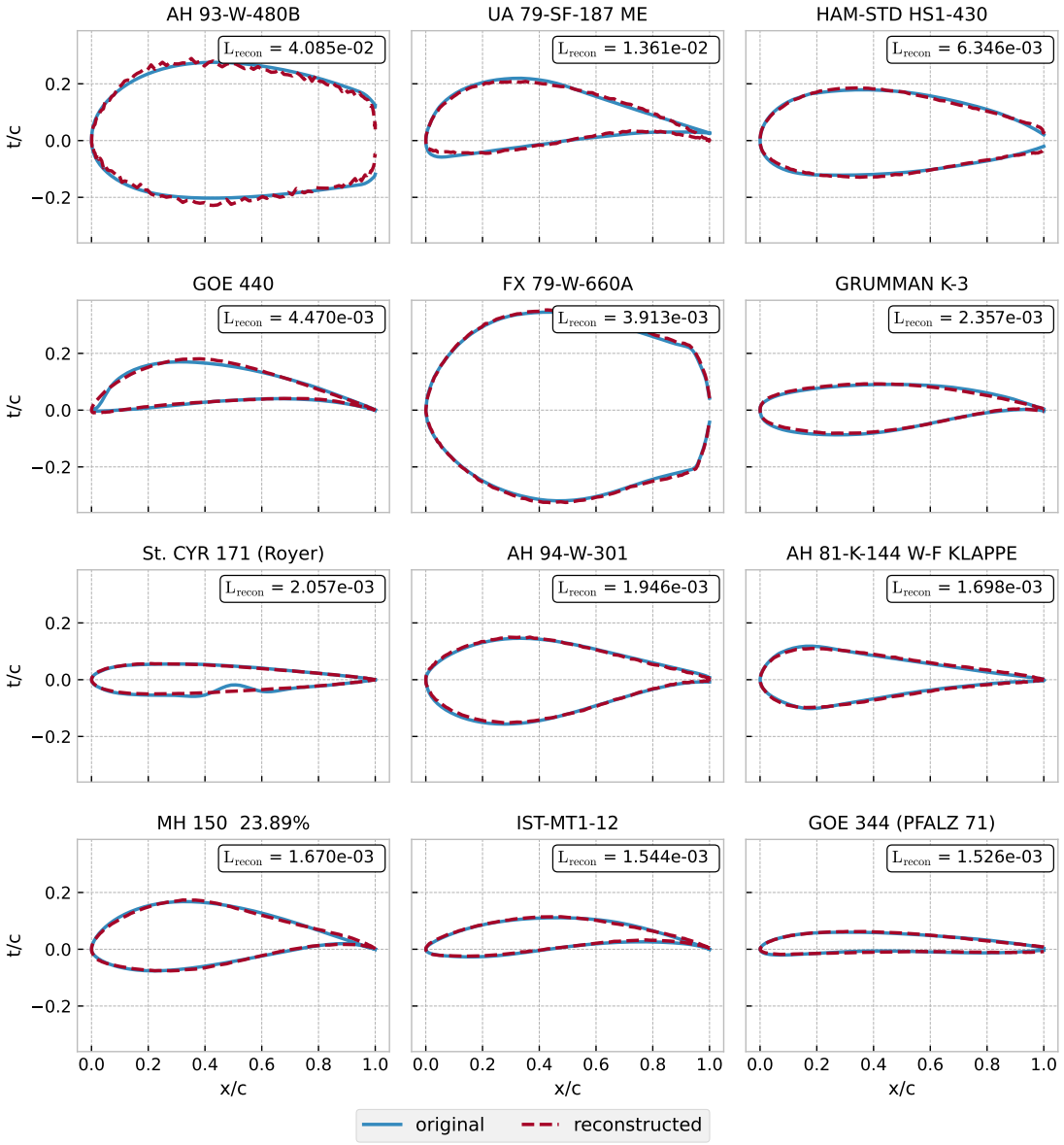

Fig. 5 Validation plots showing the reconstruction of the 12 airfoils with the highest reconstruction loss. Reconstruction loss values are marked in the corner of each plot.

standard deviations away from its mean, $Z_i = \mu_i \pm 2 \cdot \sigma_i$, while all other latent parameters are kept at their mean values $Z_{j \neq i} = \mu_j$. The resulting airfoils are compared in Figure 7.

To further explore the impact of each latent parameter on specific airfoil characteristics, a series of airfoil properties, collectively denoted as $\mathcal{P}$, are calculated from the generated airfoils. These properties encompass a range of geometric features critical to airfoil design and performance:

- Maximum thickness ($t_{\max}$) and its location ($x_{t_{\max}}$): represents the greatest distance between the upper and lower surfaces of the airfoil.
- Maximum camber ($c_{\max}$) and its location ($x_{c_{\max}}$): corresponds to the highest y-coordinate of the camber line, indicating the maximum curvature of the airfoil.
- Leading edge radius (R_{LE}): defined as the inverse of the curvature at the leading edge of the airfoil.
- Leading edge angle (θ_{LE}): the angle formed by a vector extending from the leading edge to the point at $x = 0.02$ on the camber line.

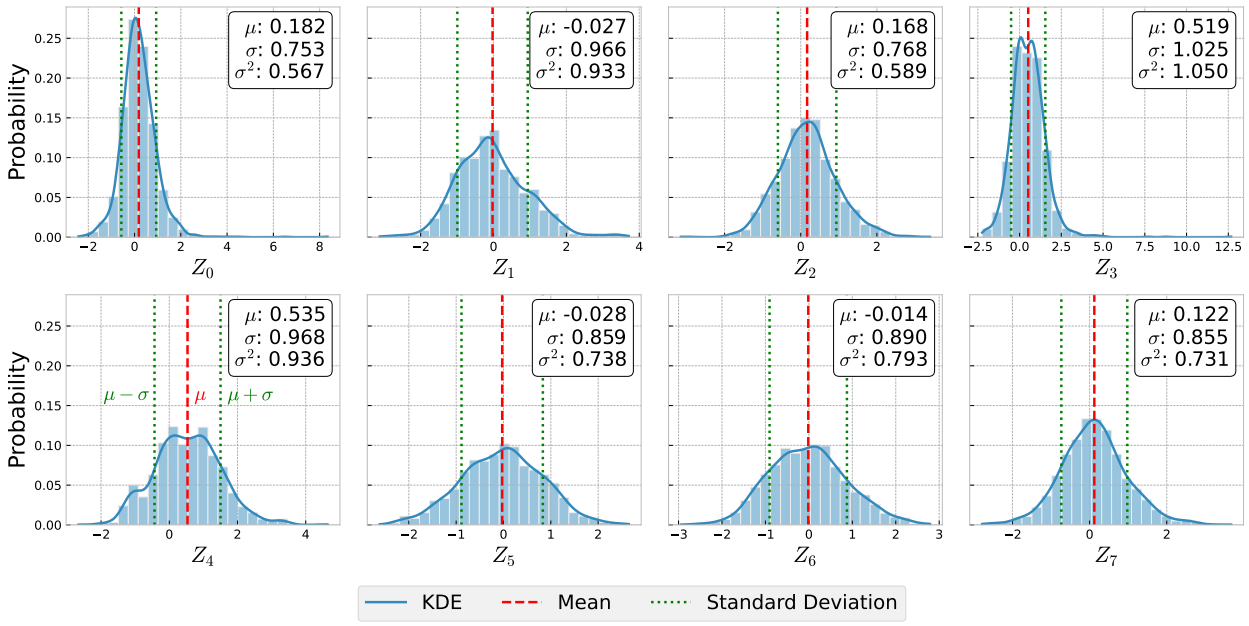

Fig. 6 Distribution of the latent parameters.

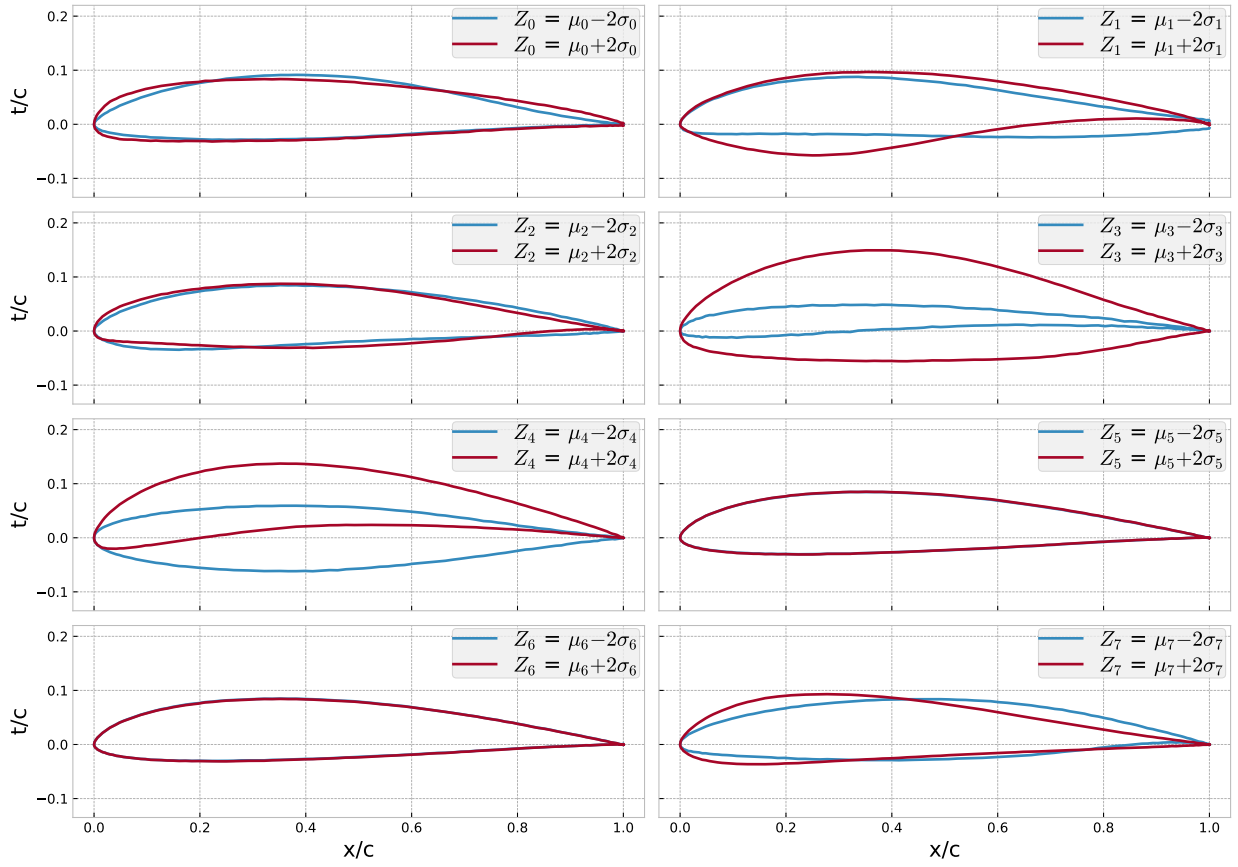

Fig. 7 Effect of the latent parameters on the generated shape. Each parameter is set to $Z_i = \mu_i \pm 2\sigma_i$ while the rest are kept at their mean values $Z_{j\neq i} = \mu_j$.

- Trailing edge angle (θ_{TE}): the angle formed by a vector from the trailing edge to the point at $x = 0.98$ on the camber line.
- Trailing edge wedge angle (γ_{TE}): the angle between vectors drawn from the endpoints of the upper and lower surfaces to $x = 0.95$, representing the wedge shape at the trailing edge.

The trailing edge is defined as the midpoint between the final points on the upper and lower airfoil surface. The leading edge is then the point at the furthest distance from the trailing edge point.

To evaluate how each latent parameter affects the airfoil shape, each latent parameter is varied within two standard deviations from its mean value, as presented in Equation 8, , while keeping all other latent parameters at their respective mean values, as defined by Equation 9. This approach allows for an isolated assessment of each latent parameter's influence on the airfoil properties.

$$Z_i \in [\mu_i + 2 \cdot \sigma_i,\ \mu_i - 2 \cdot \sigma_i] \tag{8}$$

$$Z_{j \neq i} = \mu_j \tag{9}$$

To contextualize and compare the effects of the latent parameters on the airfoil properties, a normalization process is applied. Each measured property $\mathcal{P}$ is normalized with respect to its total value range and the largest relative change ($\tilde{\delta}_{max_i}$) observed for that property, as defined by Equation 10. The maximum relative change δ_{max_i} of a property $\mathcal{P}$, under the influence of the latent parameter Z_i that most significantly impacts that property, is determined using Equation 11. Here, $\mathcal{P}_i$ represents the values of the property $\mathcal{P}$ under the influence of the i^{th} latent parameter, while $\mathcal{P}$ represents all recorded values of a property across all latent parameter variations. Finally, the obtained normalized property values $\tilde{\mathcal{P}}_i$ are plotted in latent traversal plots shown in Figure 8. This normalization facilitates understanding how each latent parameter influences the airfoil shape relative to other latent parameters, and how each property varies relatively to the other properties.

$$\tilde{\mathcal{P}}_i = \frac{\mathcal{P}_i - \mathcal{P}_{i_{\text{min}}}}{[\max(\mathcal{P} - \min(\mathcal{P})] \cdot \tilde{\delta}_{\max_i}} \tag{10}$$

$$\tilde{\delta}_{max_i} = \max \left[\frac{\max(\mathcal{P}_i) - \min(\mathcal{P}_i)}{\max(\mathcal{P}) - \min(\mathcal{P})} \right] \tag{11}$$

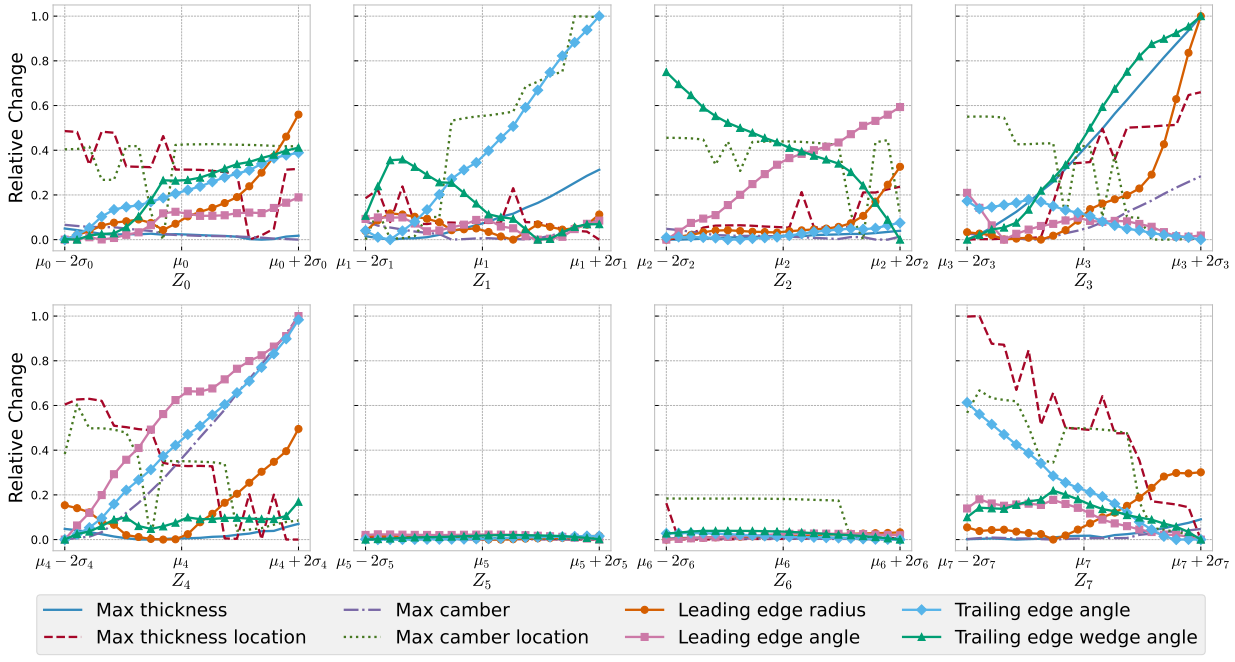

Fig. 8 Latent traversal plots showing the change in normalized property values for each latent parameter. Unaltered latent parameters are kept at their mean value $Z_{j \neq i} = \mu_j$.

What should be observed in the traversal plots is a strong correlation for some of the parameters, signifying that the corresponding latent parameter directly controls that property. A linear relationship indicates a clear, predictable change in the airfoil property as the latent parameter varies, reflecting a well-defined control mechanism within the latent space. In addition to traversal plots, another effective method for visualizing the effects of latent parameters on airfoil properties is through a correlation matrix, as illustrated in Figure 9. This matrix highlights the correlations between latent parameters and airfoil properties, echoing the observations made in the traversal plots. In this matrix, a correlation coefficient with an absolute value greater than 0.5 can be considered significant, indicating a strong relationship. Coefficients with lower values might still denote correlation, but could also be influenced by noise in the data. To refine the analysis, the correlation coefficients are scaled relative to the magnitude of the impact that latent parameters have on the airfoil properties. This scaling involves normalizing the range of each property relative to its maximum value for each latent parameter (Equation 12), followed by normalizing these relative differences between 0 and 1 (Equation 13), which yields the property correlation weights w_i for each latent parameter Z_i. The resulting scaled correlation coefficients provide a more nuanced view of the relationships, taking into account the magnitude of each latent parameter's influence.

$$\tilde{\delta}_i = \frac{\max(\mathcal{P}_i) - \min(\mathcal{P}_i)}{\max(\mathcal{P}_i)} \tag{12}$$

$$w_i = \frac{\tilde{\delta}_i - \min(\tilde{\delta}_i)}{\max(\tilde{\delta}_i) - \min(\tilde{\delta}_i)} \tag{13}$$

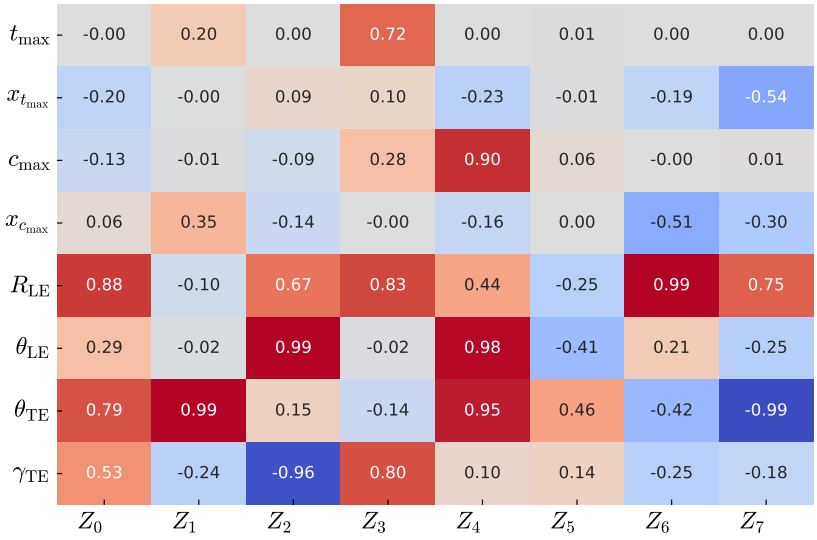

Fig. 9 Weighted latent-property correlation matrix. Latent parameters Z_i are varied one at a time while the rest is kept at their respective mean value $Z_{j\neq i} = \mu_j$

Despite this normalization and scaling, some parameters may still appear more influential than they are in reality. For instance, the leading edge radius and trailing edge wedge angle demonstrate a stronger correlation with Z_3, although the airfoil plots (Figure 7) clearly show the maximum thickness varying the most. This discrepancy might stem from these properties undergoing more significant magnitude changes than thickness, even after normalization. In the future, a different approach to normalization and scaling might yield a more accurate representation of the relationships. Nevertheless, with its current normalization and scaling the correlation matrix provides valuable insights into how latent parameters are related to airfoil properties. It provides a more precise reflection of the influence of each latent parameter, enhancing the interpretability and utility of the latent space in airfoil design and optimization.

The visualizations, including the latent traversal plots and the correlation matrix, provide detailed insight in how each latent parameter influences specific airfoil properties. These insights help in decoding the latent space's structure and the relationships between latent parameters and airfoil characteristics.

- Z_0 predominantly affects the leading edge radius and trailing edge wedge angle. The correlation matrix shows it also has a noticeable influence on the trailing edge angle. This suggests that Z_0 is key in defining the airfoil's edge characteristics.
- Z_1 strongly correlates to the trailing edge angle, but also shows clear influence on the maximum camber location and maximum thickness. It gives the camber line an S-shape, varying both the strength and direction of its curves. From an aerodynamics point of view, this corresponds with a more forward or aft loading of the airfoil.
- Z_2 mainly governs the leading edge angle and inversely affects the trailing edge angles. Visually, this parameter has a more significant role in shaping the airfoil's leading edge profile.
- Z_3 clearly controls the maximum thickness of the airfoil. Since these properties are inherently linked, also the trailing edge wedge angle, leading edge radius, and, to a smaller degree, the maximum camber follow along.
- Z_4 has captured the maximum camber value of the airfoil, which in turn varies the leading and trailing edge angles.
- Z_5 and Z_6 do not appear to have captured any features, indicating these may be redundant and simply produce random Gaussian noise under the influence of the KL divergence, and effectively get filtered out by the generator network.
- Z_7 mostly varies the location of the maximum thickness and camber location when visually inspecting the generated airfoils and looking at the traversal plots. However, since these properties are subject to sudden shifts due to roughness in the surfaces, they get overshadowed by the strong correlations of the leading edge radius and trailing edge angle.

Oscillating lines in the traversal plots indicate no direct control of a particular property. However, it is important to note that properties like maximum thickness location and maximum camber location can vary due to local irregularities in the airfoil's geometry. Such variations might not necessarily reflect a direct influence of the latent parameters but rather the inherent non-smoothness in the airfoil shapes. On the other hand, they also make it harder to recognize correlation, as was the case for latent parameter Z_7, where clear changes in the plots ended up with small correlation in the matrix due to the nonlinearity of the changes caused by jumps in the value.

The observation that a single latent parameter affects multiple airfoil properties might initially seem to indicate poor disentanglement. However, these connections are logical, reflecting inherent relationships in airfoil geometry. For instance, an increase in airfoil thickness typically results in a larger trailing edge wedge angle, and variations in maximum camber affect the leading and trailing edge angles. To illustrate these relationships, Figure 10 presents a correlation matrix of the airfoil properties. This matrix visually confirms the logical links between properties, such as the correlation between maximum thickness and leading edge radius and wedge angle, and the synchronized movement of maximum camber and thickness locations. This analysis shows that the interdependencies among airfoil characteristics in the VAE's latent space represent realistic geometrical relationships, providing valuable insights for airfoil design and optimization.

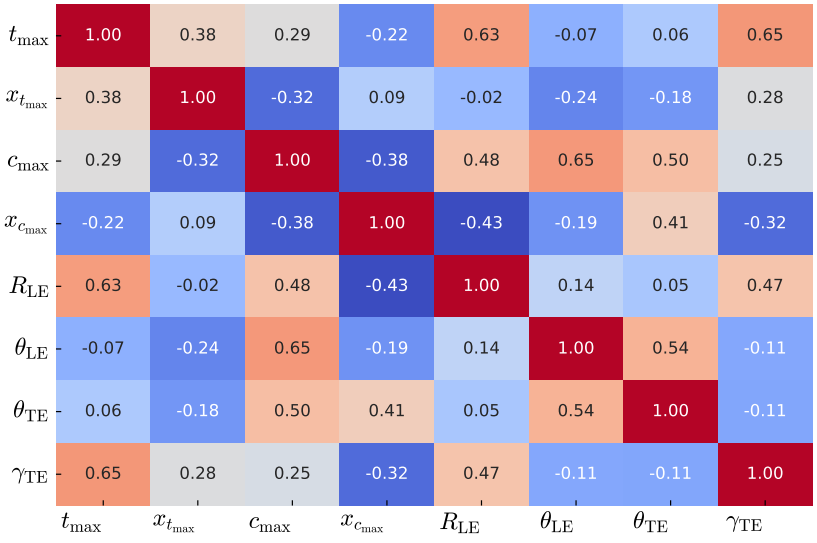

Fig. 10 Airfoil property correlation matrix shows which of the measured airfoil properties are inherently linked.

In the previous analysis, each latent parameter Z_i was varied individually while keeping the other parameters fixed at their mean values, $Z_{j\neq i} = \mu_j$. To further validate the findings regarding the effects of latent parameters on airfoil properties, a more dynamic approach is employed. In this approach, while one latent parameter is altered according to Equation 14, the remaining parameters are not held constant but are instead randomly sampled. Specifically, the unaltered latent parameters are sampled from $Z_{j\neq i} \sim \mu_j \pm 2 \cdot \sigma_j$, using a variation of the parameterization trick employing a uniform distribution $\mathcal{U}(0, 1)$, as shown in Equation 15.

$$Z_i \in [\mu_i + 2 \cdot \sigma_i,\ \mu_i - 2 \cdot \sigma_i] \tag{14}$$

$$Z_{j\neq i} = \mu_j + \varepsilon \cdot 2\sigma_j, \quad \text{where } \varepsilon \sim \mathcal{U}(0, 1) \tag{15}$$

The latent traversal plots and correlation matrix are then recreated under these new conditions, with the results presented in Figure 11 and Figure 12. Analysis of these updated visualizations reveals that the correlations between latent parameters and airfoil properties remain largely unchanged. Moreover, the influence of a particular latent parameter on the generated airfoil is consistent, regardless of the variations in the other latent parameters.

This consistency in the correlations and effects, even under the condition of randomly sampled unaltered latent parameters, underscores the robustness of the observed relationships. It confirms that the effects attributed to each latent parameter are not artifacts of keeping other parameters constant but are inherent properties of the VAE's latent space. This finding reinforces the reliability of the VAE model in capturing and manipulating specific airfoil characteristics and provides further assurance in its application for airfoil design and optimization tasks.

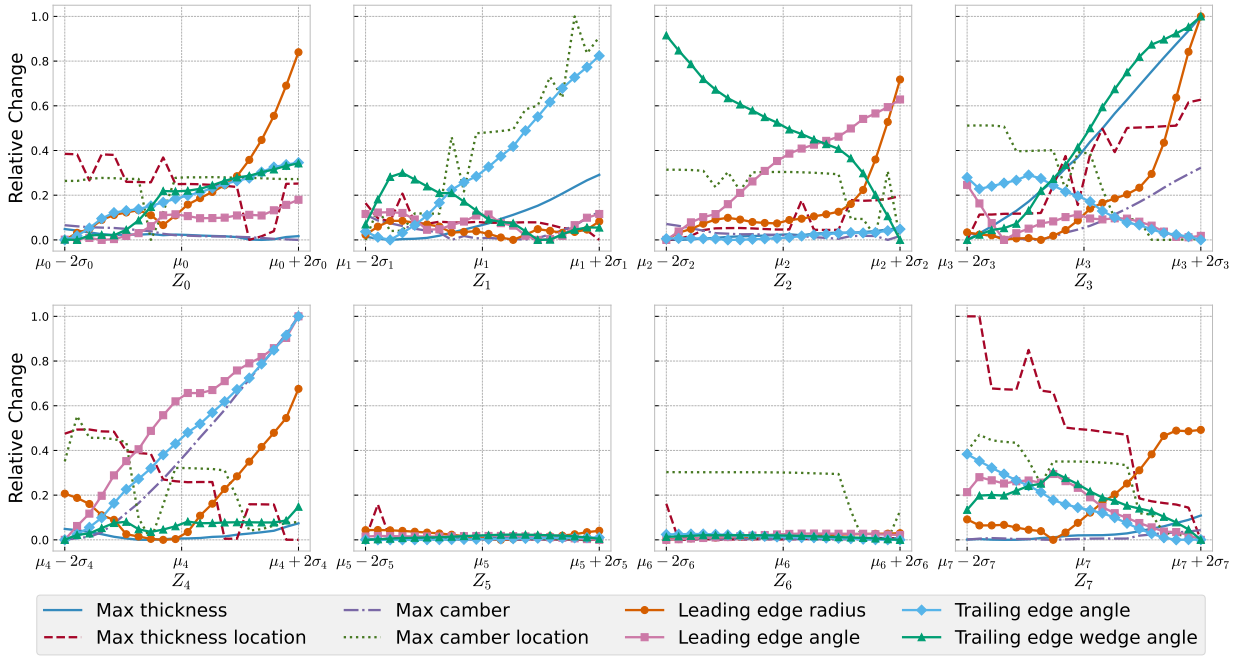

Fig. 11 Latent traversal plots showing the change in normalized property values for each latent parameter, where unchanged latent parameter values are sampled randomly from within two standard deviations $Z_{j\neq i} \sim \mu_j \pm 2 \cdot \sigma_j$.

VIII. Conclusions and future work

The training of a variational autoencoder network for interpretable airfoil parameterization has been successfully conducted. The investigation into the disentanglement of the latent space, through various visual and numerical methods, suggests that effective parameterization of airfoils in the UIUC database is possible with just six latent parameters. Although some of the airfoil properties used for correlation analysis were interrelated, the network distinctly identified key airfoil features such as maximum thickness, maximum camber, and the location of maximum thickness. Additional parameters were found to influence the leading edge radius, trailing edge wedge angle, leading edge angle, and the curvature shape of the camberline, potentially affecting the airfoil's aerodynamic load distribution. Incorporating

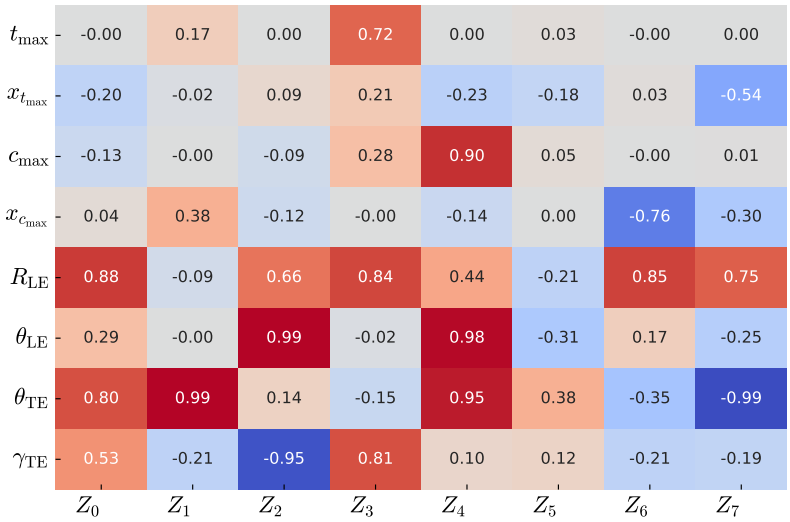

Fig. 12 Latent-property correlation matrix, where unchanged latent parameter values are sampled randomly from within two standard deviations $Z_{j\neq i} \sim \mu_j \pm 2 \cdot \sigma_j$.

aerodynamic analysis into this framework could introduce a new spectrum of properties for evaluation, further enriching the model's capability to characterize and generate airfoil shapes based on performance criteria. This integration could significantly expand the utility and application scope of the developed model. The results could potentially be enhanced through more meticulous preprocessing of the training data. The current dataset contains inconsistencies, such as sparse coordinate files, errors, and varying orientations and trailing edge gaps. Development of a tool for more rigorous data processing is underway. Nevertheless, even with the existing dataset, the network effectively provides a low-dimensional, interpretable representation of airfoils. The encoder and generator networks derived from this model facilitate quick parameterization of airfoils from coordinates, and the generation of new airfoils using the identified feature variables. Further improvements might include using a broader range of airfoil properties for constructing correlation matrices, providing deeper insights into the network's feature capturing capabilities. Refining normalization and scaling methods for traversal plots and correlation matrices could lead to more precise correlations between latent parameters and airfoil geometry changes. An intriguing possibility is to develop a metric for disentanglement based on correlation matrices, offering a single-value assessment of the latent space's interpretability. However, this approach would need careful consideration, as the choice of parameters for correlation analysis could significantly influence the outcome. Future research areas are rich and varied, encompassing the exploration of different network architectures, loss functions, and methods to enhance data preprocessing. These investigations will not only refine the current model but also expand the understanding and applications of variational autoencoders in airfoil design and beyond.

References

[1] Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks Are Universal Approximators," Neural Networks, Vol. 2, No. 5, 1989, pp. 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

[2] Li, J., Du, X., and Martins, J. R., "Machine Learning in Aerodynamic Shape Optimization," Progress in Aerospace Sciences, Vol. 134, 2022, p. 100849. https://doi.org/10.1016/j.paerosci.2022.100849.

[3] Le Clainche, S., Ferrer, E., Gibson, S., Cross, E., Parente, A., and Vinuesa, R., "Improving Aircraft Performance Using Machine Learning: A Review," Aerospace Science and Technology, Vol. 138, 2023, p. 108354. https://doi.org/10.1016/j.ast.2023.108354.

[4] Regenwetter, L., Nobari, A. H., and Ahmed, F., "Deep Generative Models in Engineering Design: A Review," , Mar. 2022.

[5] Wang, J., He, C., Li, R., Chen, H., Zhai, C., and Zhang, M., "Flow Field Prediction of Supercritical Airfoils via Variational Autoencoder Based Deep Learning Framework," Physics of Fluids, Vol. 33, No. 8, 2021, p. 086108. https://doi.org/10.1063/5.0053979.

[6] Saetta, E., Tognaccini, R., and Iaccarino, G., "AbbottAE: An Autoencoder for Airfoil Aerodynamics," AIAA AVIATION 2023 Forum, American Institute of Aeronautics and Astronautics, San Diego, CA and Online, 2023. https://doi.org/10.2514/6.2023-4364.

[7] Du, X., He, P., and Martins, J. R. R. A., "A B-Spline-based Generative Adversarial Network Model for Fast Interactive Airfoil Aerodynamic Optimization," AIAA Scitech 2020 Forum, American Institute of Aeronautics and Astronautics, Orlando, FL, 2020. https://doi.org/10.2514/6.2020-2128.

[8] Wang, Y., Deng, L., Wan, Y., Yang, Z., Yang, W., Chen, C., Zhao, D., Wang, F., and Guo, Y., "An Intelligent Method for Predicting the Pressure Coefficient Curve of Airfoil-Based Conditional Generative Adversarial Networks," IEEE Transactions on Neural Networks and Learning Systems, 2021, pp. 1–15. https://doi.org/10.1109/TNNLS.2021.3111911.

[9] Du, X., He, P., and Martins, J. R., "Rapid Airfoil Design Optimization via Neural Networks-Based Parameterization and Surrogate Modeling," Aerospace Science and Technology, Vol. 113, 2021, p. 106701. https://doi.org/10.1016/j.ast.2021.106701.

[10] Kou, J., Botero-Bolívar, L., Ballano, R., Marino, O., de Santana, L., Valero, E., and Ferrer, E., "Aeroacoustic Airfoil Shape Optimization Enhanced by Autoencoders," , Sep. 2022.

[11] Yonekura, K., and Suzuki, K., "Data-Driven Design Exploration Method Using Conditional Variational Autoencoder for Airfoil Design," Structural and Multidisciplinary Optimization, Vol. 64, No. 2, 2021, pp. 613–624. https://doi.org/10.1007/s00158-021-02851-0.

[12] Yang, Z., Jiang, H., and Lan, Z., "3D Conceptual Design Using Deep Learning," 2018. https://doi.org/10.48550/ARXIV.1808.01675.

[13] Higgins, I., Matthey, L., Glorot, X., Pal, A., Uria, B., Blundell, C., Mohamed, S., and Lerchner, A., "Early Visual Concept Learning with Unsupervised Deep Learning," 2016. https://doi.org/10.48550/ARXIV.1606.05579.

[14] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A., "Generative Adversarial Networks: An Overview," IEEE Signal Processing Magazine, Vol. 35, No. 1, 2018, pp. 53–65. https://doi.org/10.1109/MSP.2017.2765202.

[15] Gui, J., Sun, Z., Wen, Y., Tao, D., and Ye, J., "A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications," IEEE Transactions on Knowledge and Data Engineering, 2022, pp. 1–1. https://doi.org/10.1109/TKDE.2021.3130191.

[16] Kingma, D. P., and Welling, M., "Auto-Encoding Variational Bayes," 2013. https://doi.org/10.48550/ARXIV.1312.6114.

[17] Foster, D., Generative Deep Learning, O'Reilly Media, Inc., Sebastopol, California, 2019.

[18] Goodfellow, I., Bengio, Y., and Courville, A., Deep Learning, Adaptive Computation and Machine Learning, The MIT Press, Cambridge, Massachusetts, 2016.

[19] Li, F.-F., "Deep Learning for Computer Vision (CS231n) Lecture 13: Generative Models," , May 2017.

[20] Brunton, S. L., Noack, B. R., and Koumoutsakos, P., "Machine Learning for Fluid Mechanics," Annual Review of Fluid Mechanics, Vol. 52, No. 1, 2020, pp. 477–508. https://doi.org/10.1146/annurev-fluid-010719-060214.

[21] Hinton, G. E., and Salakhutdinov, R. R., "Reducing the Dimensionality of Data with Neural Networks," Science, Vol. 313, No. 5786, 2006, pp. 504–507. https://doi.org/10.1126/science.1127647.

[22] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," , 2019.

[23] Selig, M., "UIUC Airfoil Data Site," https://m-selig.ae.illinois.edu/ads/coord_database.html, 1996.

[24] Wilson, D., and Martinez, T. R., "The general inefficiency of batch training for gradient descent learning," Neural Networks, Vol. 16, No. 10, 2003, pp. 1429–1451. https://doi.org/https://doi.org/10.1016/S0893-6080(03)00138-2, URL https://www.sciencedirect.com/science/article/pii/S0893608003001382.

[25] Karpathy, A., "A recipe for training neural networks," , 2019. URL http://karpathy.github.io/2019/04/25/recipe/.

[26] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M., "Optuna: A Next-generation Hyperparameter Optimization Framework," 2019. https://doi.org/10.48550/ARXIV.1907.10902.