University of Naples Federico II
Naples, Italy

Machine Learning in Aerodynamics: Clustering and Autoencoders for CFD Applications

Ettore Saetta

Thesis for the degree of
Doctor of Philosophy
in Industrial Engineering
Cycle XXXVI

Supervisors: Prof. Renato Tognaccini
Prof. Gianluca Iaccarino

Coordinator: Prof. Michele Grassi

October 2023

Machine Learning in Aerodynamics: Clustering and Autoencoders for CFD Applications

Abstract

A hot topic in all scientific fields over the last decade has been Machine Learning (ML), a branch of artificial intelligence. Fluid Mechanics has also been exploring the great potential of ML technology. Aerodynamic analyses traditionally involve Experimental Fluid Dynamics (EFD), Computational Fluid Dynamics (CFD), and semi-empirical models for preliminary design. In the context of CFD, the rapid growth of computational resources and development of new technologies and algorithms, is progressively increasing the feasibility to perform high-fidelity CFD simulations, even for complex geometries. Nevertheless, the journey toward the practical implementation of high-fidelity CFD for real-world applications remains challenging.

In this context ML holds the potential to revolutionize our approach to Fluid Dynamic analyses and significantly enhance their outcomes. Despite initial skepticism, ML has found successful integration in various real-world applications, ranging from autonomous vehicles to medical diagnosis. Also in the Fluid Dynamic framework, it proved to be very promising for a wide array of applications, including turbulence modeling, flow control, shape optimization, and flow field prediction.

A fascinating aspect of combining ML with Fluid Mechanics is that Fluid Mechanics is grounded in well-established theories and first principles. The fusion of physical knowledge with data-driven methodologies can be mutually beneficial for both Fluid Mechanics and ML. As Fluid Mechanics continues to uncover the great potential of ML, various concerns and open questions remain. The main areas of focus include the interpretability of these models, the handling of extensive data acquired from numerical simulations and experiments, ensuring that these models yield physically meaningful results, and the uncertainty quantification of the ML outcomes.

In this dissertation, ML algorithms are investigated within the context of CFD, with a particular focus on unsupervised algorithms. In a first part, a well-established clustering algorithm is repurposed for a novel application: physical-based domain decomposition. This is crucial for various applications, ranging from the shape optimization to the aerodynamic forces decomposition. The ML method demonstrated its ability to overcome the limitations of existing *deterministic* algorithms used to date.

In a second part, autoencoders (unsupervised deep-learning algorithms) are employed for real-time predictions of flow fields around wing sections. The investigation on autoencoders, as presented here, begins with the assessment of their accuracy in predicting aerodynamic flow fields, particularly in strongly non-linear regimes. Subsequently, it addresses several challenges in ML techniques, including the interpretability of models, by performing extreme data compression, and the assessment of the uncertainty quantification of the autoencoder predictions. This part is pivotal for enabling the deployment of ML technologies in real-world systems.

The ML algorithms investigated here are linked by a common underlying theme: the unsupervised extraction of field information and the inherent uncertainty embedded in this process.

The applications proposed here should be intended as a proof of the significant impact that ML can have on aerodynamic analyses in the near future, unlocking the possibility to perform high-fidelity CFD simulations for industrial applications within reasonable timeframes and contained computational costs. Furthermore, it could open the door to real-time flow predictions with a satisfactory level of accuracy, enabling tasks like shape optimization, real-time flow control, and visualization using high-fidelity data.

Contents

Nomenclature

Symbols

a	=	speed of sound
c	=	airfoil chord
C_D	=	drag coefficient (unit surface)
C_d	=	drag coefficient (unit length)
C_{D_i}	=	lift-induced drag coefficient (unit surface)
C_{D_v}	=	viscous drag coefficient (unit surface)
C_{D_w}	=	wave drag coefficient (unit surface)
C_L	=	lift coefficient (unit surface)
C_l	=	lift coefficient (unit length)
C_p	=	pressure coefficient
D_{far}	=	far field drag
D_{sp}	=	spurious drag
D_v	=	viscous drag
D_w	=	wave drag
$D_{\Delta s}$	=	entropy drag
$\bar{f}$	=	reduced frequency
F_{bl_ε}	=	non-dimensional boundary layer sensor (dissipation of turbulent kinetic energy)
$F_{bl_{\mu_t}}$	=	non-dimensional boundary layer sensor (eddy viscosity)
F_{bl_Φ}	=	non-dimensional boundary layer sensor (dissipation function)
F_{shock}	=	non-dimensional shock wave sensor
F^D_{shock}	=	non-dimensional Ducros shock wave sensor
h	=	normalized averaged grid cell size
k	=	turbulent kinetic energy
K	=	kernel size
$\mathcal{L}$	=	loss function
$\hat{\mathcal{L}}$	=	Maximum likelihood
m	=	maximum airfoil camber
M	=	Mach number

N_b	=	batch size
N_d	=	number of free parameters of the decoder
N_e	=	number of free parameters of the encoder
N_{ls}	=	latent space dimension
p	=	static pressure
p_m	=	position of maximum airfoil camber
P_s	=	padding size
r_i	=	relevance index
R	=	Universal gas constant
Re	=	Reynolds number
S	=	stride size
t	=	airfoil thickness
u	=	first velocity component
v	=	second velocity component
V_i	=	i-th latent variable
w	=	weight of a connection in a neural network
$\mathbf{W}$	=	matrix of weights of a neural network
a	=	angle of attack
γ	=	ratio of specific heats
Δs	=	entropy variation
ε	=	small constant value
ε	=	dissipation of turbulent kinetic energy
η	=	learning rate
λ	=	weight of the regularization term in the loss function
μ_j	=	cluster centroid
μ_l	=	laminar viscosity
μ_t	=	eddy viscosity
$\widetilde{\nu}$	=	Spalart variable
ρ	=	density
$\overline{\Phi}$	=	dissipation function of kinetic energy
Φ_t	=	disspiation function of turbulent kinetic energy
ω	=	vorticity vector
$\boldsymbol{\nabla}$	=	gradient operator

Acronyms

AE	=	Autoencoder
AI	=	Artificial Intelligence
ANN	=	Artificial Neural Network
BIC	=	Bayesian Information Criteria

CFD	=	Computational Fluid Dynamics
CNN	=	Convolutional Neural Network
CPU	=	Central Processing Unit
DNS	=	Direct Numerical Simulations
DSV	=	Dynamic Stall Vortex
EFD	=	Experimental Fluid Dynamics
EM	=	Expectation - Maximization
FLOP	=	Floating point Operations
GAN	=	Generative Adversarial Network
GCN	=	Graph Convolution Network
GMM	=	Gaussian Mixture Models
GPR	=	Gaussian Process Regression
GPU	=	Graphics Processing Unit
GS	=	Gradient Sharpening
HPC	=	High Performance Computer
LES	=	Large Eddy Simulations
LHS	=	Latin Hypercube Sampling
NN	=	Neural Networks
ML	=	Machine Learning
MLP	=	Multi Layer Perceptron
MSE	=	Mean Squared Error
POD	=	Proper Orthogonal Decomposition
RANS	=	Reynolds-Averaged Navier-Stokes
RGB	=	Red-Green-Blue
SA	=	Spalart-Allmaras
SD	=	Summed Distances
SDF	=	Signed Distance Function
SWBLI	=	Shock-Wave/Boundary-Layer Interaction
UQ	=	Uncertainty Quantification
URANS	=	Unsteady Reynolds-Averaged Navier-Stokes
VAE	=	Variational Autoencoder

Listing of figures

Listing of tables

Introduction

Aerodynamics is one of the cornerstone of aeronautical science, whose fundamental principles govern the lift, drag, and stability of aircraft, influencing their performance and maneuverability. Traditionally, the quest for new aerodynamic designs has been pursued through Experimental Fluid Dynamics (EFD), Computational Fluid Dynamics (CFD) simulations or semi-empirical models (for preliminary design purposes). These methods, while invaluable and well established, have limitations in terms of time, cost, and complexity.

In the context of CFD applications, they are computationally demanding, especially for high fidelity simulations at the Reynolds number of industrial interest ($Re\,\mathcal{O}\left(10^6\right)$). Consider, for instance, that the computational cost to perform a direct numerical simulation (DNS) scales with $\sim\ Re_{L_x}^{37/14}$ and for a wall-resolving large eddy simulation (LES) is $\sim\ Re_{L_x}^{13/7}$ [4]. An estimate of the NASA *CFD Vision 2030* says "wall-modeled LES is feasible in a 24-hour turnaround time at Reynolds number of about 1 million on unit-aspect ratio geometries using existing algorithms" [5]. This estimate is done considering the most powerful high performance computer (HPC) existing: *Tianhe-2* developed by China's National University of Defense Technology, with a peak performance of 55 PFLOP/s. This makes it impractical to carry out a high-fidelity CFD simulation for an industrially significant configuration within a limited timeframe, especially when lacking access to extensive computational resources. Beyond the simulations themselves, even using Reynolds-Averaged Navier-Stokes (RANS) [6] simulations embedded into iterative loops for shape optimization is time-consuming and computationally intensive. The challenges become even more significant when considering real-time flow predictions for active control or the creation of a digital twin of the system.

In this scenario, a transformative technology, with the potential to redefine the field of aerodynamics, has emerged: Machine Learning (ML). ML has the potential to transform our approach to comprehending, analyzing, and enhancing physical phenomena. Using advanced algorithms, neural networks, and data-driven insights, ML opens up a novel frontier for enhancing aircraft design, performance, and safety. Despite initial (and present) skepticism, ML has demonstrated to be a powerful tool for real-life applications across many different scientific fields. In the present days, the field of Fluid Mechanics is beginning to unlock the potential of employing such technologies also for enhancing fluid flows analyses and predictive capabilities. ML has historically found application in fields with a large availability of data, but limited the-

oretical knowledge. Fluid mechanics, on the other hand, has solid theories and first principles, and it is rapidly becoming rich of data. The challenge now lies in efficiently managing and leveraging this data. Now, thanks to the ML, theoretical developments are merging with data-driven analyses. "We are entering a new and exciting era in Fluid Mechanic research" [1], where this fusion has the potential to transform both Fluid Mechanics and ML.

Core subject of the present dissertation is to explore the potential of ML for aerodynamic applications. Significant attention is devoted to unsupervised ML techniques, which do not necessitate labeled data for training. Specifically, two primary challenges are addressed: a clustering problem employing the Gaussian Mixture Model and a regression problem using Autoencoders as generative networks. The overarching theme throughout the entire dissertation is the exploration of ML strategies for encoding field information, whether in discrete forms such as clusters or in continuous form within an autoencoder's latent space, while addressing the inherent uncertainties embedded in this process. The structure of the thesis is outlined as follows:

- **Chapter 1**: the history of ML is traced from its origins to the present day, highlighting key milestones along the way. An overview of the current state-of-the-art in ML as applied to Fluid Mechanics is presented. Additionally, the chapter introduces the specific contribution of this dissertation within this evolving landscape.

- **Chapter 2**: an innovative method for a physical-based domain decomposition is proposed. The Gaussian Mixture Model is employed to solve a clustering problem. This ML algorithm has the capability to address many of the limitations and difficulties associated with current *deterministic* methods. Additionally, a practical application is described for the aerodynamic drag breakdown using *far field methods*.

- **Chapter 3**: Autoencoders (AE), which are deep learning algorithms designed for dimensionality reduction and feature extraction, are thoroughly investigated for the prediction of aerodynamic fields. Initially, the study evaluates the effectiveness of AE in predicting complex aerodynamic phenomena, including airfoil static and dynamic stall. Some of the open challenges are addressed: the interpretability of ML models, and the data requirements for data-driven algorithms (fundamental if dealing with high fidelity CFD simulations).

- **Chapter 4**: the uncertainty quantification of ML models is still challenging, and quantifying uncertainty in the predictions generated by these algorithms is a crucial factor for deploying ML models into practical applications. This chapter is focused on AEs and on the effect of uncertainties due to the architecture, the hyperparamaters and the choice of the training data (internal or model-form uncertainties). Simulations accounting for internal uncertainties are also compared to the impact of the variability induced by uncertain operating conditions (external uncertainties). A comparisons with a Gaussian Process regression approach is also provided, showing the AE advantage in extracting

useful information on the prediction confidence even in the absence of ground truth data.

1

Machine Learning in Fluid Dynamics

Artificial Intelligence (AI) has a long history that began in the mid-1950s when it was established as an academic discipline in 1956. ML is a branch and integral part of the AI. Its history, as understood in the modern context, has been marked by challenges over the years. Already in 1948 A. M. Turing introduced the idea of a "learning machine" in a report titled "Intelligent Machinery" [7, 8], which discussed the concept of a machine that could modify its own instructions based on experience. This can be seen as an early exploration of ML principles. However, the term "Machine Learning" was later coined by A. Samuel in 1959 [9] to describe the field of computer science that focuses on creating systems that can learn and improve from data and experience.

The real breakthroughs in this field can be traced back to 1958, when Rosenblatt [10] showed a first example of a very simple neural network with learning capabilities: the "perceptron". The perceptron was designed as a simplified model of a biological neuron, with inputs, weights, and an activation function, to capture some of the basic principles of neural processing. It became the prototype for modern Artificial Neural Networks (ANNs). The perceptron was used for binary classification tasks, trying to separate data points into two categories based on the input features. It is worth to describe the functioning of the first perceptron, as it serves as the foundational building block for "modern" ANNs.

The perceptron is composed by one layer which takes a set of binary inputs (typically 0 or 1) or real-valued numbers. Each input is associated with a weight, which can be thought of as

a measure of the input's importance in the decision-making process. The inputs are then multiplied by their corresponding weights. The weighted sum is calculated by taking the dot product of the input vector and the weight vector (linear combination of the inputs and weights). The weighted sum is then passed through an activation function. In Rosenblatt's original work, the used activation function was a simple step function (returning 0 or 1 based on whether the weighted sum was lower or greater than a certain threshold). The output of the activation function is used for binary classification decision. This simple process can be summarized by the following formula:

$$Y = \varphi(\mathbf{X} \cdot \mathbf{W}) \tag{1.1}$$

where Y is the output (0 or 1); $\mathbf{X}$ is the input vector; $\mathbf{W}$ is the weight vector and φ is the step activation function. The perceptron is trained using the so-called "perceptron learning rule" algorithm, which updates the weights directly based on errors made by the perceptron, adjusting them to minimize classification errors.

A notable limitation of this approach was pointed out in a book by Minsky and Papert [11]: the perceptron can only model linearly separable functions and is not able to learn the logical function XOR (exclusive OR), which represents a non-linear problem, but the perceptron is a linear classifier. In order to learn the XOR function, a multi-layer perceptron (a simple fully connected neural network) is necessary, but the computational resources of that time played as show stopper. This book marked the onset of what is known as the "AI winter", where the decreased interest in perceptrons was accompanied by a broader decline in interest in AI in general. In 1974, the so-called Lighthill report [12] strongly criticized ML technologies, concluding that AI research had not made significant advancements, pointing out the field's inability to fulfill the high expectations that had been set for it. The report cited various reasons for the perceived stagnation in AI research, including the absence of robust theoretical foundations, the inability to scale up from toy problems to real-world applications, and an excessive reliance on heuristic methods. The result of this report was a reduced interest and investment in AI research in United Kingdom, followed by the United States. However, despite these challenges, the study of more complex learning algorithms continued to make progress.

A new blooming of AI was registered in 1980s, with the rediscovery of the *backpropagation* algorithm [13]. Rumelhart et al. published an experimental analysis of the technique [14]. This contribution played a crucial role in popularizing backpropagation and kick-started a dynamic era of research in multilayer perceptrons. This breakthrough facilitated the training of neural networks (NN) with multiple layers. In 1989, Hornik et al. in [15] developed the Universal approximation theorem. They rigorously proved that standard multilayer feedforward networks with at least one hidden layer, using arbitrary activation functions, are capable of approximating any Borel measurable function [16]. In this regard, multilayer feedforward networks are considered a class of universal approximators. Despite initial enthusiasm, the actual successes in ML fell short of expectations, leading (again) to reduced investments in the field in the early 1990s.

However, in the mid-1990s, an interest of studying NN was growing, and the field of ML

shifted towards embracing more statistical approaches. Noteworthy, in this period, are the development of *Random Forest Algorithm* [17], *Support-Vector Machines* [18] and *Recurrent Neural Networks* [19], which marked significant breakthroughs in the field of ML research.

The early 21st century witnessed a significant surge in research and a renewed interest in the field of ML. This resurgence can be attributed to the convergence of three key trends. Firstly, the consistent growth in the volume of collected data has ushered in the era of *Big Data*. Secondly, advancements in computational resources (such as GPUs) have reached a level of computing power that is no longer a hindrance. Lastly, the development of new algorithms for deep ML has played a pivotal role in driving this renewed interest in the field. In 2000, a significant milestone was achieved with the establishment of the *Journal of Machine Learning Research*, which has since become one of the foremost journals in the field of ML. In the last two decades, ML started to integrate into the lives of people and industries. Its applications became increasingly prevalent in various fields, including social media, transportation, natural language processing, healthcare, and finance. Since the idea of *deep learning* (deep NNs, composed by multiple hidden layers) was first theorized in the 1980s, it is only in the last two decades that much of the research is centered around this idea. In the present day, ML technologies have matured to a point where numerous companies worldwide are effectively integrating ML algorithms into their products, such as computer vision for self-driving vehicles, face recognition and medical diagnosis; natural language processing for digital assistants and translators; and many other applications in every field.

1.1 An overview on ML

ML algorithms can be divided into three main categories as reported in Fig. 1.1.1.

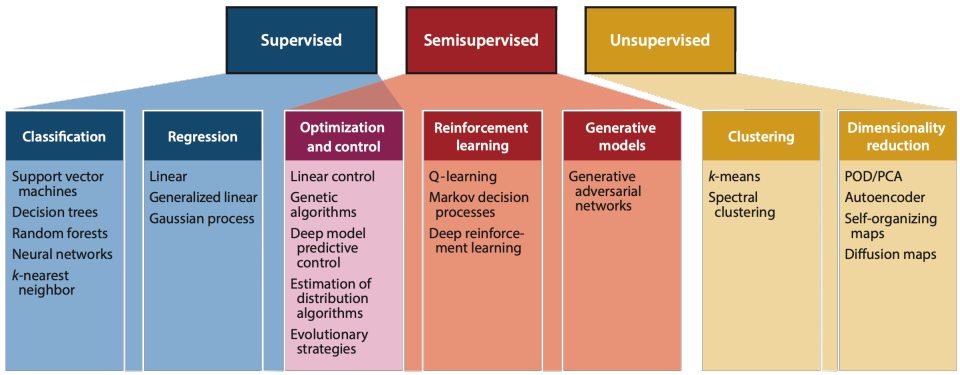


Figure 1.1.1: ML taxonomy. Extracted from [1].

- **Supervised** algorithms are the most adopted. They require databases composed by labeled data for classification and regression tasks. The learning algorithm is trained pro-

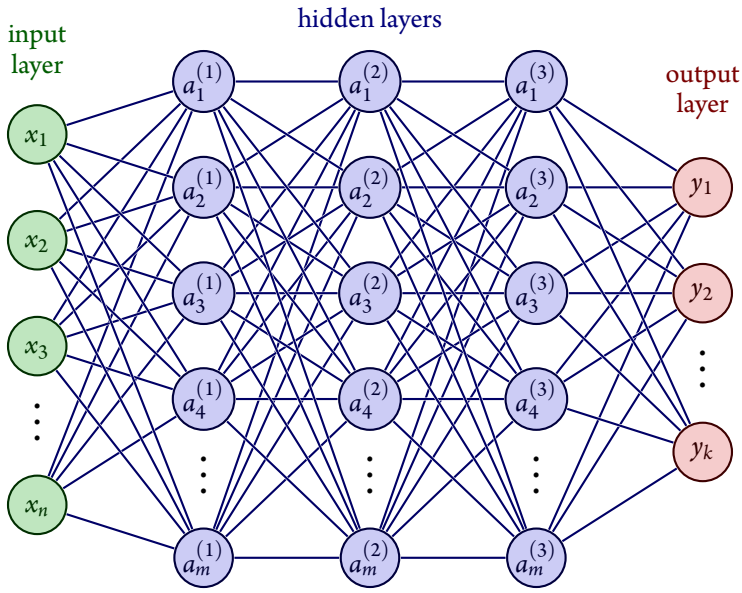


Figure 1.1.2: Schematic representation of a fully connected neural network.

viding a *ground truth* composed by input and corresponding output. After the training phase, the algorithm should be able to predict the output of a new "unseen" input. The most common algorithms for this category are NNs (briefly described below) for solving classification or regression problems and the k-nearest neighbor [20].

- **Unsupervised** algorithms are mainly employed for solving clustering and dimensionality reduction problems. The training dataset is not labeled, but the algorithm is aimed to extract features and patterns from data. Most widespread algorithms are the k-means for clustering and Autoencoders for dimensionality reduction and feature extraction. The latter, being the primary focus of this study, is extensively discussed in the following sections.
- **Semisupervised** algorithms are something in the between: there is a partial supervision and generally the algorithms are trained through processes based on game theory. In this context, reinforcement learning and generative adversarial networks take the lead.

1.1.1 Neural Networks

It is worth to briefly introduce the functioning of a simple NN, which underlies the operation of many ML algorithms. Figure 1.1.2 shows a schematic example of a fully connected NN composed by 1 input layer, 3 hidden layers and 1 output layer. Lets assume $\mathbf{x} \in \mathbb{R}^n$ is an input vector of dimension n and $\mathbf{y} \in \mathbb{R}^k$ is the output vector of dimension k. Each hidden layer is composed by m units named *neurons* and denoted with $a_j^{(l)}$, where l and j are indices denoting the layer and

the neuron respectively. In a fully connected NN, each neuron is connected with all the neurons of the previous and the next layers (there are no inter-layer connections). Each connection represent a *weight* and can be denoted with $w_{i,j}$ where i and j are respectively the indices of the two connected layers by the weight $w_{i,j}$ (the i-th neuron of the previous layer and the j-th neuron of the present layer). For each j-th neuron in a layer l, the weighted sum of the input is computed and a bias b is added:

$$z_j^{(l)} = \sum_i w_{i,j}^{(l-1)} a_i^{(l-1)} + b_j^{(l-1)} \tag{1.2}$$

this is then passed into a non-linear activation function $\sigma(\cdot)$ to introduce non-linearity into the model:

$$a_j^{(l)} = \sigma\left(z_j^{(l)}\right) \tag{1.3}$$

In vectorial form, the output of a layer can be written as:

$$a^{(l)} = \sigma\left(\mathbf{W}^{(l-1)} a^{(l-1)} + \mathbf{b}^{(l-1)}\right) \tag{1.4}$$

where $\mathbf{W}$ is the weight matrix and $\mathbf{b}$ the bias vector. The procedure is repeated for all the hidden layers until the output layer.

Many activation functions can be employed to introduce non-linearity in the NN. The most widely used functions are the *Rectified Linear Unit* (ReLU):

$$\text{ReLU}(x) = (x)^+ = \max(0,\ x) = \frac{x - |x|}{2} \tag{1.5}$$

and the Sigmoid:

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{1.6}$$

The choice of the activation function constitutes a hyperparameter in the NN architecture that must be made with care. It is crucial to select the appropriate activation function for the output layer as well. For instance, when solving a binary classification problem, one may prefer to employ a Sigmoid function or a *binary step function*, which is restricted to the interval $[0,\ 1]$ to aid in training. Conversely, when tackling a regression problem, one may opt to utilize an identity function which is not restricted. A comprehensive examination of various activation functions can be found in [21].

Convolutional Neural Networks (CNNs) have a fundamental interest in this dissertation. They are NNs where at least one layer performs a convolution operation. Convolutions avoid the complexity of the dense connectivity present in full connected layers and lead to sparse interactions that are more appropriate when the inputs are representing continuous data; furthermore convolutions lead to equivariant representations that are critical to extract large-scale patterns present in the data [22]. A convolution operation (denoted with $*$), in discrete form, is

given by:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)\, w(t-a) \tag{1.7}$$

where x is the input, w is the kernel and s is the feature map (the result of the convolution operation). Figure 1.1.3 illustrates a schematic example of the convolution operation in a CNN.

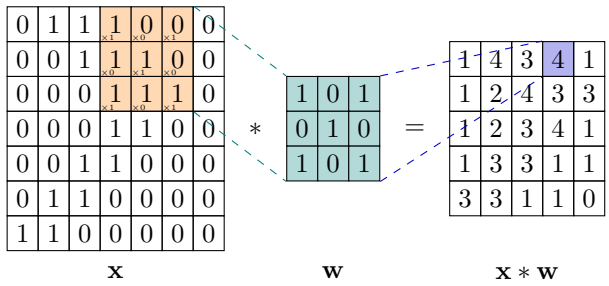


Figure 1.1.3: Schematic representation of a convolution operation in a CNN.

Convolution effectively reduces the dimensionality of the input while extracting and manipulating the most critical features (information). The elements of the kernel matrix are *learned* during the training phase of a CNN, granting the training process the flexibility to determine the most suitable kernels for achieving the objective of minimizing the loss function. Conversely, the kernel size and stride are hyperparameters selected by the user. Specifically, the *stride* denotes the step size taken by the convolutional kernel as it moves across the input data. A larger stride can be beneficial for downsampling the input, reducing the computational complexity of the network, but it may also lead to information loss.

The dimension of the intermediate outputs of convolutional layers can be derived by the following formula:

$$\begin{bmatrix} n \\ m \end{bmatrix}^{(i)} = \frac{1}{S} \begin{bmatrix} n \\ m \end{bmatrix}^{(i-1)} + \frac{2P_s - K + S}{S} \tag{1.8}$$

where $[n\ m]^{(i)}$ is the dimension of the i-th intermediate output; $[n\ m]^{(i-1)}$ the dimension of the previous output (the input to the i-th layer); K, P_s and S are the Kernel size, the Padding size and the Stride size respectively. To simplify the notation, no superscript is adopted on K, P_s and S, which are referred to the i-th convolutional layer. If they have different dimensions in the two directions, then they become vectors. Similarly, from eq. (1.8) it is possible to derive the formula for computing the dimension of the intermediate output in the transpose convolutions of the decoder:

$$\begin{bmatrix} n \\ m \end{bmatrix}^{(i)} = S \begin{bmatrix} n \\ m \end{bmatrix}^{(i-1)} - S + K - 2P_s \tag{1.9}$$

The channel dimension c is an hyperparameter of the architecture which can be chosen without

any particular constraint.

The weights and the biases of the network are selected during the *training* phase. In this stage, an optimization algorithm is employed to updates the weights of the network, by minimizing an objective function (*loss function* $\mathcal{L}$, which is typically the error between the output of the NN and the ground truth). The most common loss function employed in this framework is:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{1.10}$$

which corresponds to the Mean Squared Error (MSE) between the output of the NN y and the ground truth $\hat{y}$. Many different loss functions can be employed depending on the specific task. Further details of the employed $\mathcal{L}$ will be presented later in the dissertation.

Each iteration of the optimizer is called *epoch*, and during each epoch, for each weight in the NN, the gradient of the loss with respect to the weights is computed. Using the gradient information, the weights are updated in the opposite direction of the gradient to minimize the loss:

$$w_{i,j}^{(l)} = w_{i,j}^{(l)} - \eta\frac{\partial\mathcal{L}}{\partial w_{i,j}^{(l)}} \tag{1.11}$$

where η is one of the hyperparameters of the training algorithm, named *learning rate*. The optimizer keeps on updating the weights, until a stop criterion is satisfied. This procedure is tipically performed using the gradient descent [23] and the back-propagation algorithm [14]. The former is the optimization algorithm which uses the gradients to minimize the loss function; the latter calculates these gradients from the last, back to the first layer, using the chain rule.

The database utilized for training a NN requires meticulous preparation. One common practice is to normalize the dataset for various reasons, including enhancing numerical stability during optimization. Each practitioner may employ their own normalization approach, tailoring it to the specific data characteristics and NN architecture. The most prevalent technique, as utilized in this dissertation, is referred to as *min-max scaling*, which involves rescaling the dataset to a range between 0 and 1.

Regarding the database construction, it is initially divided into two parts: one designated as the *training set* and the other as the *test set*. From the training set, a smaller portion is extracted and designated as the *validation set*. The differentiation between these three sub-sets is as follows: the *training set* is employed for the primary purpose of effectively training the NN, enabling the adjustment of trainable parameters (weights and biases); the *validation set* serves a distinct role during the training process, as it is employed to monitor the quality of training convergence. Notably, it does not contribute to updating the weights of the NN but plays a pivotal role in fine-tuning user-defined hyperparameters; the *test set* is reserved for use after the training phase is complete. It is employed to assess the NN's performance in terms of generalization to new, unseen data. The training set can be presented to the NN during the training phase in three distinct manners, each defining a different type of training:

- Batch Gradient Descent: in this approach, the entire training dataset is considered to compute a single step of the optimizer within an epoch. In other words, for each epoch, there is only one update of the training parameters, made for the whole training set.
- Stochastic Gradient Descent: this method involves considering only one data point (of the training set) at a time to compute a single step of the optimizer. In each epoch the weights are updated as many times as the number of data points in the training set.
- Mini Batch Gradient Descent: with this technique, the training database is divided into batches, and each optimization step is carried out on one batch at a time. In each epoch the weights of the NN are updated for each batch. The size of the batches is a user-defined training hyperparameter.

In this dissertation, it's noted that Mini Batch Gradient Descent is the chosen approach for all the computations. For a more comprehensive exploration of various gradient descent techniques, further details can be found in [24].

Once the network is trained, it can be employed for *online* use to make predictions of new unseen cases. This is just a simple example of an artificial NN to introduce the nomenclature that is adopted in this work.

When discussing ML, NNs are often the first to come to mind. However, as partially illustrated in Figure 1.1.1, there is a wide array of NN architectures and ML algorithms at disposal. The specific ML algorithms adopted in this work, are detailed in their respective chapters.

1.2 Applications in Fluid Dynamics

In the last two decades, the ML overwhelmed also the Fluid Dynamic field. Higher computing powers and the availability of a large amount of data and public databases [25] made it feasible to adopt ML technologies. Indeed, Fluid Mechanics deal with massive amount of data generated by EFD or CFD. These data are feeding public databases available to everybody leading to the necessity of finding new efficient ways to use and analyze Fluid Mechanic problems [1, 26]. Generally, ML is employed in fields characterized by abundant data but a scarcity of theoretical knowledge. Fluid Mechanics is rich of both data and first principles, but ML methodology can give rise to new challenges and developments in both fields: it is possible to have a better understanding of data using theories and to develop new theories extracted from data.

Most ML applications in Fluid Dynamics were primarily focused on tasks such as turbulence modeling, optimization and control of fluids. NNs proved to be a powerful tool to develop new data-driven turbulence models [26–29]. In particular, it is possible to train a NN using data from experiments, LES or DNS, to develop a turbulence model without the need of calibrated parameters for RANS simulations. NNs can also be used for optimization purposes in order to improve performance and efficiency, drastically reducing the need for CFD calls [30, 31].

Fluid Dynamic analyses deal with Navier-Stokes equations, which are beyond our current computational resources for high Reynolds number flow and complex geometries of industrial interest. For this reason, simulations are performed using approximations and models of some of the terms of the Navier-Stokes equations. Reynolds-Averaged Navier-Stokes (RANS) are the most adopted for industrial applications to retrieve the overall aerodynamic characteristics of a geometry of interest. Nonetheless, traditional RANS and Unsteady RANS (URANS) turbulence models are often inadequate for accurate predictions of a steady mean flow or capturing its essential attributes. These models struggle to address critical features, including inherent three-dimensionality and unsteadiness, which are of great significance in applications related, for instance, to Aeroacoustics and Aeroelasticity. LES, DNS or EFD need to be employed. However, such approaches are often too expensive to be integrated into optimization iterative loops or for real-time flow control. For these reasons, reduced-order models and dimensionality reduction algorithms have been developed and employed for a wide range of applications. In this framework, ML technologies identify a favorable setting.

Attributing all ideas used in ANNs can be somewhat challenging due to the diverse literature and evolving terminology over time [32]. However, the initial applications of ML technologies to flow-related issues trace their origins back to the early 1990s. These applications primarily emerged in the context of EFD, particularly in the domain of Particle Image Velocimetry [33, 34]. While, one of the first application of deep learning technologies (as intended today) in a Fluid Dynamic setting is attributed to Milano and Koumoutsakos [28], who showed the equivalence between proper orthogonal decomposition (POD) and linear NNs for reconstructing the near-wall field in a turbulent flow by leveraging the flow fields obtained from DNS. They also proposed a non-linear NN which provided improved reconstruction and prediction capabilities for the near wall velocity fields.

Within the vast body of research that combines ML technologies with Fluid Dynamic applications, several noteworthy articles have been published. Recent advancements in turbulence modeling and the integration of ML with CFD solvers are discussed, with a primary focus on the application of autoencoders for generative purposes. It should be noted that other notable applications, such as physics-informed learning [35, 36], shape optimization [37], flow control [38], and EFD [39, 40] will not be addressed in more details in this dissertation due to their deviation from the central focus.

1.2.1 Turbulence modeling

As previously mentioned, DNS is often unfeasible for many real-world applications due to the considerable computational expenses associated with resolving all scales, particularly for flows with high Reynolds numbers. Additionally, challenges emerge when dealing with complex geometries. Hence, industrial CFD today still rely on RANS simulations, where no turbulent scales are simulated, or LES which resolve only the larger turbulence scales while modeling the smaller ones. RANS models are inadequate when dealing with strong pressure-gradients, highly

unsteady and separated flows, as well as significant three-dimensional effects. Consequently, one of the most prevalent applications of ML for CFD involves the development of turbulence models for RANS equations using neural NN. A wide range of ML algorithms have been developed to improve the modeling of RANS turbulence models. Noteworthy is the work by Tracey et al. [27], where they employ supervised learning algorithms to create a representation of turbulence modeling closure terms. Notably, they offer a proof-of-concept for the development of new turbulence models using ANNs. The proof lies in their training of an ANN to model the source terms from the Spalart-Allmaras (SA) turbulence model, and embedded the model into a RANS solver. This work underscore the promise and feasibility of ML approaches in facilitating turbulence model development. Later, Ling et al. [41], trained a NN to model the Reynolds stress tensor, using high-fidelity simulations. They proposed a novel NN architecture designed to introduce Galileian invariance into the predicted anisotropy tensor, leading to improved outcomes when compared to the baseline RANS simulation. In this wake, a countless number of papers on turbulence modeling have appeared in the literature. An interesting review focusing on this branch is provided by Duraisamy et al. in [26].

1.2.2 ML and CFD solver integration

Highly intriguing applications include the integration of ML models into CFD solvers to enhance computational performance. In this framework, Obiols-Sales et al. [42] present the so-called *CFDNet*, where a CFD solver performs few starting iterations on a given problem (namely *warmup*), then a CNN inferring the steady state and finally an iterative refinement stage to correct the solution is performed by the CFD solver. In their experiments, they achieved a speed-up ranging from 1.9x to 7.4x compared to the time required for a full CFD simulation. In this context, noteworthy is the work by de Avila Belbute-Peres et al. [43], who employed a graph convolution network (GCN) coupled with a CFD solver (SU2 [44]). They conducted experiments to explore various methods of integrating the GCN with the CFD solver. Ultimately, they adopted a hybrid approach in which a coarse mesh was provided to the CFD solver and a fine mesh to the GCN. After a specific number of iterations, the coarse mesh containing the CFD solution was fed into the GCN for upsampling, followed by additional GCN iterations.

1.2.3 Surrogate models

ML in the context of Fluid Dynamics, also finds application in flow predictions, which is the primary focus of this dissertation. In this framework, ANNs are trained to provide the flow predictions (or quantity of interest) for a wide range of Fluid Dynamic problems. The ML models are then used for the fast prediction of new unseen cases. In this framework, generative networks are employed as surrogate models. Also in this case, a wide range of different NN arhitectures and methodologies can be found in literature. For instance, a simple ML model, is proposed by Zhang et al. in [45], where giving in input the image of airfoils to a CNN, they were able to

accurately predict the aerodynamic force and moment coefficients at different Reynolds number, Mach number and angle of attack. A more complex methodology is proposed by Sekar et al. [46], whose objective was the prediction of the entire incompressible laminar steady flow field over airfoils. Their method is based on a combination of CNN and multi layer perceptron (MLP). The former is adopted to extract geometrical parameters form the airfoil; the latter to generate the flow field using these geometrical parameters, the Reynolds number and the angle of attack. The method they proposed is able to predict an accurate flow field of new geometries in seconds. Wu et al. [47] proposed a generative adversarial network (GAN) combined with CNNs to accurately predict the flow field of supercritical airfoils. They only consider geometrical variations in their dataset, by fixing Reynolds number and Mach number.

Of particular relevance for the present dissertation are the implementations of Autoencoder (AE) architectures within CFD applications. Noteworthy applications include the work of Agostini [48], which focuses on the flow behind a cylinder. The study demonstrates that combinations of an AE with other ML algorithms can serve multiple purposes, such as offering a low-dimensional dynamical model, providing deterministic flow predictions, and recovering high-resolution spatio-temporal data from contaminated or under-sampled data. One of its outcomes involves constructing a dynamical model within the latent space of an AE, achieved by applying a clustering algorithm directly to the latent space. He also provides comparison between the performance of the AE and a conventional method, frequently used in Fluid Mechanics: the POD. He shows the superior performance achievable with the AE approach. In this context, he presents the AE as a non-linear generalization of the POD. Bhatnagar et al. in [49] showed the AE capability to reproduce the flow around airfoils with different shapes, angle of attack and Reynolds number. In their work they used a signed distance function (SDF) as input to the encoder and provided the velocity and pressure fields as output of the decoder. In this way it is possible to obtain the flow field around a new geometry in real-time by using only geometrical information (the SDF). The novelty of their work lies in comparing two different architectures: a conventional AE, and an encoder with three separate decoders, each predicting one of the output physical variables, with and without gradient sharpening (GS) in the loss function [50]. They concluded that the best performance was achieved using a conventional AE architecture (with a shared encoder-decoder), while incorporating GS to enforce sharp reconstructions. Tsangali et al. [51] showed an interesting analysis on the generalizability of AE for flow field predictions. They used a massive database composed by 11400 RANS simulations encompassing various airfoil geometries, angle of attacks, Reynolds numbers, and Mach numbers. The range of variations for Re_∞ spanned from 10^2 to 3×10^6, and for M_∞ from 0.2 to 0.7, ensuring that the dataset covered a wide array of physically distinct flow regimes. In their application of the AE, it was employed in a *supervised* manner. Instead, the input to the AE consisted of the signed distance function (SDF) and the free-stream information, while the output of the AE is the whole flow field in terms of density and velocity components. This article is of interest because it recognizes the need for an extensive database to enable an AE to accurately generate

flow fields across a wide range of physical regimes. Additionally, it requires a fully connected latent space composed by 2800 elements. The practice of employing supervised AE as black-box models with large datasets and high-dimensional latent spaces is prevalent in the literature.

A different approach was taken by Kang et al. in their recent work [52]. They developed ROMs using both AE and β-variational autoencoders (β-VAE) [53, 54] to reproduce the transonic flow around a RAE2822 airfoil at various Mach numbers and angles of attack. Notably, they employed a smaller dataset consisting of 500 cases and restricted the latent space dimension to a range from 2 to 16 latent variables. Furthermore, they carried out a performance evaluation, comparing a physics-aware β-VAE with a physics-unaware β-VAE. They shoewd improved performance when incorporating physical information into the latent space of a β-VAE. Li et al. in [55] proposed a VAE for the inverse design of supercritical airfoils, injecting physical information (free-stream Mach number, lift coefficient, shock wave position and Mach number upstream the shock wave) in the latent space of the AE. This approach assumes the physical parameters necessary to describe the phenomenon are known a-priori and provided to the AE during the training phase.

In the past decade, the number of papers on ML applications in Fluid Mechanics has seen exponential growth. While the full potential of this technology has not yet been fully realized in the field of Fluid Mechanics, ongoing developments and discoveries continue to emerge in both areas. This points to a promising future for ML in Fluid Mechanic applications.

1.3 Present contribution

The objective of this dissertation is to explore the potential and limitations of unsupervised ML algorithms for aerodynamic applications.

A first part of this work is dedicated to applying a well-known clustering algorithm, the Gaussian Mixture Model (a ML algorithm), in a novel context. Specifically, it is employed for physical-based domain decomposition in a compressible viscous flow at high Reynolds numbers. Traditionally, domain decomposition relies on computing the value of a non-dimensional sensor in each grid cell of the domain. Then, a threshold is applied, and cells with values exceeding this threshold are assigned to a specific region. This methodology has various complexities and difficulties, as discussed in section 2.2.1. In this context, ML demonstrates its ability to overcome the limitations of the current *deterministic* methods, identifying viscous, shock wave and external inviscid regions automatically, without the need for human intervention. An example of practical application is also proposed: the aerodynamic drag breakdown using *far field methods*. This work illustrates the potential of deploying ML algorithms in Fluid Mechanic applications to overcome current method limitations. It also opens up possibilities for embedding region identification algorithms within iterative optimization loops more effectively.

The second part of the dissertation goes beyond simply using AEs; it investigates some of the open questions associated with these algorithms. It places particular emphasis on the in-

terpretability of such models. Additionally, it explores methods to reduce the amount of data required for training the model. A fundamental aspect of this part is the uncertainty quantification (UQ) in ML models, which plays a pivotal role in the deployment of data-driven algorithms in real-life applications.

Unlike most of the work in literature, which use high-dimensional latent spaces, that lack interpretability, present study employs very aggressive data compression, with only 1 to 5 latent variables. The aim is to gain an interpretation of the embedded space *learned* by an AE. Interpretability of such models remains an open question, and understanding what information the encoder extracts and how it does so is crucial for a better understanding of how these algorithms work, thereby increasing confidence in their applications. In this context, it will be showed that AEs are undeniably capable of extracting physical information from the input data, a phenomenon referred to as *physical imprinting*. The imprinting is demonstrated by constructing low-dimensional databases that only encompass simple geometrical changes, and by limiting the study to very few latent variables.

Another concern in the ML community is that such algorithms are data-hungry. This is not a real problem in many fields such as image processing and computer vision, where there is abundance of data and the cost and time for collecting new data is very contained. However, in Fluid Dynamics, acquiring high-fidelity data is not only expensive but also storage occupying. Some studies in the literature tend to overlook the expenses associated with data acquisition when comparing ML algorithms with traditional methods. To address this issue, based on the physical imprinting, this work proposes an iterative algorithm for an adaptive database generation. The primary goal is to leverage targeted, albeit coarse, databases capable of delivering accurate flow predictions even when trained on small datasets. The algorithm autonomously identifies new data to enhance the database without requiring user intervention. This capability allows its application in contexts where it might be difficult to identify which data are essential for the training process.

Finally, UQ is a critical aspect when it comes to assessing the reliability of predictions made by ML algorithms. However, UQ for data-driven models remains an area with limited coverage in the literature. While many studies propose their own methods for quantifying the uncertainty of ML models, some challenges persist. For instance, evaluating ML models for their ability to acknowledge uncertainty when dealing with shifted datasets or data scarcity remains an ongoing issue. These aspects are essential for the safe integration of ML models in real-world applications. This work addresses this problem by demonstrating that ensemble UQ methods, when used with AE in a real-world application (the prediction of aerodynamic characteristics of airfoil sections), enable the development of models capable of identifying data gaps in the training datasets and appropriately recognizing uncertainty in such regions. The UQ methodology presented here strives to establish a comprehensive *protocol* for handling uncertainty in ML models. It takes into account various sources of uncertainty, including aleatory and epistemic factors, sensitivity to dataset size, data shifts, and input uncertainties. The proposed approach provides

predictions with confidence intervals that correctly track the relative importance of various uncertainty sources. Additionally, this method is compared with a more traditional approach: Gaussian Process Regression (GPR). It is illustrated the advantage of AEs in extracting useful information on the prediction confidence even in the absence of ground truth data.

2

Physical-Based Domain Decomposition

The interest in the physical-based domain decomposition appears in different fields: from the optimization of aerodynamic geometries to the aerodynamic forces breakdown. The identification of the physical region simply based on the knowledge of the thermo-fluid dynamic properties in a point is not a trivial task. Lanzetta et al. [56] showed some of the techniques adopted by the classical methods, based on the identification of proper *sensors* which, however, require the adoption of an arbitrary case-dependent threshold (cut-off) value. Moreover, they require a deep user experience and are not sufficiently robust to avoid a *visual* inspection by the user. These techniques will be referred to as *deterministic* methods. In this framework, the objective is to assess the capability of a ML algorithm in surpassing the current limitations and challenges within this field. Within the realm of ML algorithms, attention is directed towards unsupervised algorithms. They enable the extraction of features from data without a reliance on pre-defined ground-truth labels for the outcomes. For example, when addressing a *clustering* problem, homogeneous zones within the feature space can be identified solely through the analysis of data, obviating the necessity for prior user supervision.

A growing interest in clustering methods can be found in many different fields. For instance, Callaham et al. [57] described a data-driven method to identify dominant balance regimes in complex physical systems for a wide range of different applications, such as turbulence, combustion, nonlinear optics, geophysical fluids, and neuroscience. They introduce the concept of equation space, where the feature space in input to the clustering algorithm is composed by the

terms of the governing equations of the specific problem. In particular, one of the applications they propose is the identification of different regions in a DNS of a transitional flat plate by using the Gaussian Mixture algorithm. An objective global measure of the fit of dominant balances to observations is proposed by Kaiser et al. [58], who showed a method for the automatic identification of dominant balances by different ML algorithms. Their method permits objective comparison of different algorithms of balance identification, obtaining an objective framework for dominant balance identification. Other applications of clustering algorithms can be found also in Earth science, for instance Sonnewald et al. [59] used a clustering algorithm (k-means) to reveal global ocean dynamics.

In computer science applications, Patel and Kushwaha [60] adopted clustering algorithms (k-means and Gaussian Mixture) to analyze cloud workloads in terms of CPU and memory usage of Google Cluster and Bitbrains, in order to extract useful information about the workload characteristics for the enhancement of cloud services. The authors also provide an interesting comparison between the performance of the two most widespread clustering algorithms, k-means and Gaussian Mixture.

2.1 Clustering algorithms

Clustering algorithms are ML unsupervised techniques to find patterns in a data distribution. Just "looking" at the data, a clustering algorithm is able to find homogeneous zones in the feature space without using a training phase. It can be viewed as the unsupervised counterpart of classification algorithms, although it **differs significantly** from them. In unsupervised learning, the model is trained without the use of labeled data.

Two primary clustering algorithms are discussed here: the k-means which is the most simple clustering algorithm; and the Gaussian Mixture Model (GMM), which is the one selected for the analyses presented later in this work.

The input data for the clustering algorithm consists of selected variables (called *features*) for each data point, forming a matrix with dimensions equal to the number of data points and the number of features associated with each data point. The data are then represented in the *feature space*, where the axes correspond to the selected features. Both clustering algorithms, described in this context, start from this representation and differ in how they form clusters based on this feature space.

2.1.1 K-means

K-means [61, 62] is the most simple and widespread clustering algorithm. It is a *hard* clustering algorithm, meaning it categorically assigns labels to data points. This algorithm can be adopted in cases of spherical cluster shapes.

The algorithm split a set $\mathbf{X}$ of N samples in C clusters. Each j-th cluster is identified by the mean (μ_j) of the samples in the cluster. In this framework, the means are called *centroids* of the

clusters. Once the k-means model is trained, a new data point is assigned to the cluster whose centroid is closer to it in the feature space. The training of the model consists in an iterative algorithm, which objective is the minimization of the *summed distances* (*SD*), also called *inertia*:

$$SD = \sum_{i=0}^{n} \min_{\mu_j \in C} \left(||x_i - \mu_j||^2 \right) \tag{2.1}$$

where x_i is the position in the feature space of the i-th data point. The steps of the training algorithm are:

1. Initialization: the centroids are randomly chosen between the data points in the feature space.
2. Assignment: each data point is assigned to the cluster whose centroid is the closest to it (using the Euclidean distance for instance).
3. Centroid Update: each centroid is recalculated as the mean of all data points assigned to that cluster.
4. Iterations: steps 2 and 3 are repeated until *SD* goes under a tolerance value or a limit number of iterations is reached.

The number of clusters is a user-defined input, so it needs to be chosen based on the specific problem. There are methods and convergence criteria available to help select the optimal number of clusters for a given problem. Insights of such methods are discussed in section 2.2.2. Figure 2.1.1 provides a graphical illustration of the steps involved in training the k-means algorithm, summarizing the described process. In the figure, data points are represented as white circles in a two-dimensional feature space, while the three colored triangles depict the centroids. When data are assigned to a specific cluster, they are colored with the same color of the centroid. This example demonstrates a straightforward scenario in which the clusters are clearly distinct. However, in practice, this is not always the case. For real-world applications, the k-means algorithm may not converge (depending on the initialization), necessitating a restart of the training procedure. Typically, multiple restarts are conducted, and the one resulting in the lowest value of the *SD* is retained.

2.1.2 Gaussian Mixture

GMM is a *soft* clustering algorithm: it does not assign membership to a specific cluster but rather provides a probability of belonging to a cluster. Those making it more flexible and can handle cases where data points may belong to multiple clusters simultaneously. GMM is based on the idea that the data distribution in the feature space can be described by a Gaussian Mixture, where each cluster is a different Gaussian distribution composing the mixture. The data are assigned to a particular cluster using the *Expectation - Maximization* (EM) iterative algorithm

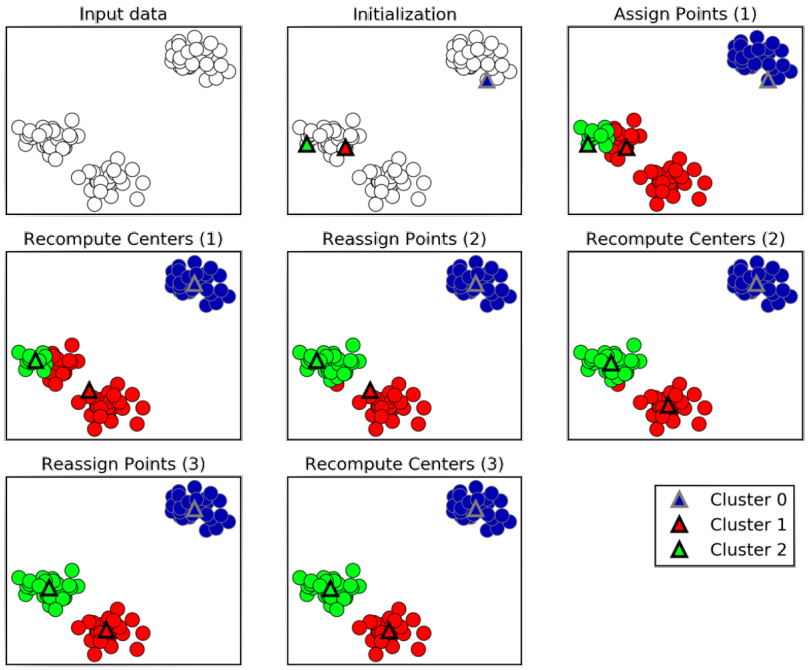


Figure 2.1.1: Example of the k-means iterative fitting procedure in a two-dimensional feature space. Image sourced from [2].

which is able to find the maximum-likelihood estimate of the parameters of the Gaussian distribution from a given data set [63]. In this way the algorithm evaluates the probability that a particular data belongs to a certain cluster.

Unlike k-means, GMM considers not only the mean of data to create clusters but also their variance. This allows GMM to handle more complex data distributions with flexible cluster shapes, while k-means assumes clusters to be spherical and equally sized.

Mathematical details of the GMM and EM algorithm are reported in Appendix A.

2.2 Physical-Based Domain Decomposition

The problem at hand pertains to the identification of homogeneous regions in a flow domain arising around a body obtained by CFD or EFD using a clustering algorithm. This problem is fundamental in many fields, for instance, Jeong & Hussaini developed the Q-criterion [64], which is a widespread algorithm for finding vortices in a flow domain. Wu et al. [65] had a similar objective in using a self organizing map, a class of unsupervised ML, to identify coherent turbulent structures in a DNS of a flow around a flat plate. Present goal is however different: given a numerical or experimental compressible viscous flow at high Reynolds numbers around a body with a complex geometry, to automatically determine if a particular point of the domain belongs

to a viscous region (boundary layer and body wake), a shock wave region or to the external inviscid region without any additional information about the domain topology and without the need for any input *ad hoc* threshold.

2.2.1 Deterministic algorithms

Typically, deterministic methods are based on the computation in each grid cell of a non-dimensional sensor F related with the region to be identified. If $F > t$, where t is an input cut-off value, the grid cell is assigned to the specified region.

Boundary layer

A number of sensors have been discussed and tested by Lanzetta et al. [56]. The interest in sensors for identifying different flow field regions, is felt also in the recent years, as discussed by Qiao et al. [66], who used the domain decomposition to perform the far field drag breakdown.

The natural local quantity discriminating the presence of a viscous region is the dissipation function (of kinetic energy). In case of a RANS solution:

$$F_{bl_\Phi} = \overline{\Phi} + \Phi_t = 2(\mu_l + \mu_t)\,(\boldsymbol{\nabla} V)^s_0 : (\boldsymbol{\nabla} V)^s_0 \tag{2.2}$$

where μ_l and μ_t are the molecular and eddy viscosities and $(\boldsymbol{\nabla} V)^s_0$ is the symmetric deviatoric part of velocity gradient. An example of the selection of the viscous region using this sensor, is proposed in Figure 2.2.1 in case of an airfoil in subsonic regime. The coordinates are non-dimensionalized with respect to the airfoil chord. All the RANS simulations are performed using open-source CFD software SU2 (Stanford University Unstructured) [44] with the k-ω SST turbulence model and the ROE convective numerical scheme. In what follows, a C-grid provided by NASA was employed*. The picture clearly shows that the selection is unsatisfactory. Indeed, the adopted sensor seems to correctly detect the boundary layer on most of the body, but not in the wake. In addition, around the leading edge, two lobes are selected which is clearly an unphysical result. This is given by a locally significant dissipation of internal energy, which arises due to the *spurious* viscosity introduced by the numerical scheme.

The selection improves adopting a sensor based on the local eddy viscosity:

$$F_{bl_{\mu_t}} = \frac{\mu_l + \mu_t}{\mu_l} \tag{2.3}$$

However, the laminar part of the boundary layer and the viscous sublayer, where $\mu_t \to 0$, cannot be identified. The problem can be solved, but it requires the knowledge of topological information of the grid.

*https://turbmodels.larc.nasa.gov/naca0012_grids.html

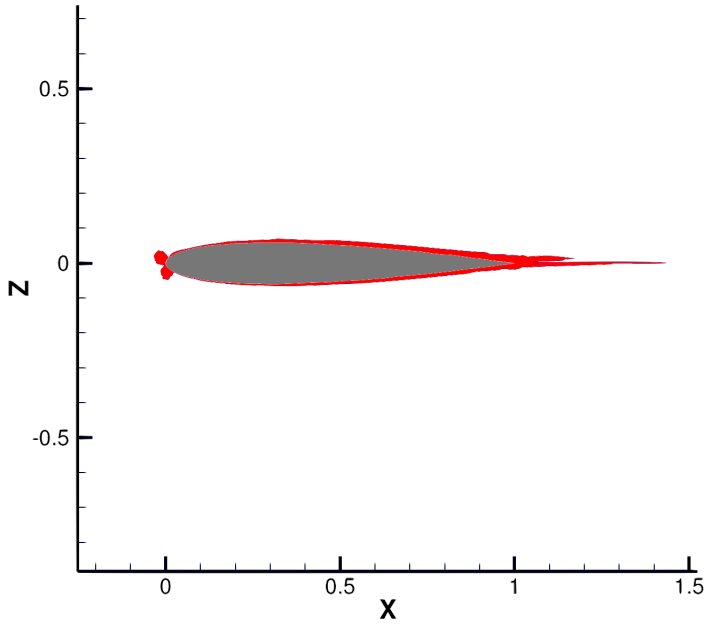


Figure 2.2.1: Selection of the viscous region by F_{bl_Φ}. NACA 0012, $M_\infty = 0.3$, $Re_\infty = 10 \times 10^6$, $a = 4°$, SST turbulence model. Cut-off value $t = 30$. Red: viscous region.

The best deterministic sensor selected by Lanzetta et al. [56] is based on the dissipation of turbulent kinetic energy ε:

$$F_{bl_\varepsilon} = \rho\varepsilon \tag{2.4}$$

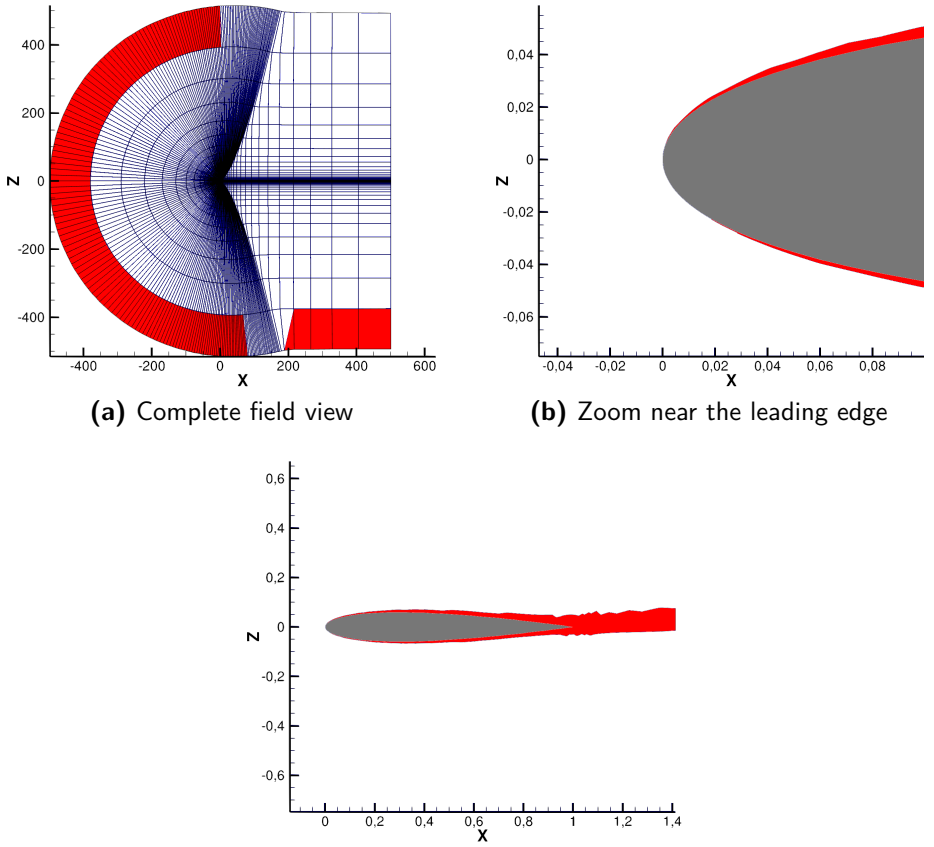
The boundary layer selection effectively improves, but in addition to the need of a cut-off parameter, another problem arises: the necessity to remove the cells selected in the far field, which again requires the knowledge of topological information (2.2.2a). Moreover, as showed in Figure 2.2.2b, the boundary layer at the leading edge of the airfoil is not selected.

Shock wave region

The identification of shock waves in a CFD solution is a problem widely discussed in literature, and particularly felt in the definition of correct shock capturing numerical methods. A first sensor, introduced by Lovely and Haimes [67], is given by:

$$F_{shock} = \frac{V \cdot \boldsymbol{\nabla} p}{a\,|\boldsymbol{\nabla} p|} \tag{2.5}$$

It represents a guess of the local Mach number orthogonal to the shock as reported by Paparone and Tognaccini [68]. The cut-off value t depends on the normal Mach number upstream of the shock, typically for transonic airfoils it is $t \approx 0.8 - 0.9$. This sensor proved to be quite robust;



(a) Complete field view

(b) Zoom near the leading edge

(c) Airfoil

Figure 2.2.2: Selection of the boundary layer by F_{bl_ε}. NACA 0012, $M_\infty = 0.3$, $Re_\infty = 10 \times 10^6$, $a = 4°$, SST turbulence model. Cut-off value $t = 0.95$. Red: viscous region.

however if more than one shock wave are present in the flow field, the choice of the cut-off value is difficult.

Another widely adopted sensor is defined by Ducros et al. [69]:

$$F^{D}_{shock} = \frac{(\boldsymbol{\nabla} \cdot V)^2}{(\boldsymbol{\nabla} \cdot V)^2 + |\omega|^2 + \varepsilon} \tag{2.6}$$

where $|\omega|$ is the vorticity magnitude and ε is a small constant value used to prevent division by zero.

Many other sensors have been developed to detect discontinuities in the flow, also involving differential operators, for more details a partial review is done by Wu et al. [70].

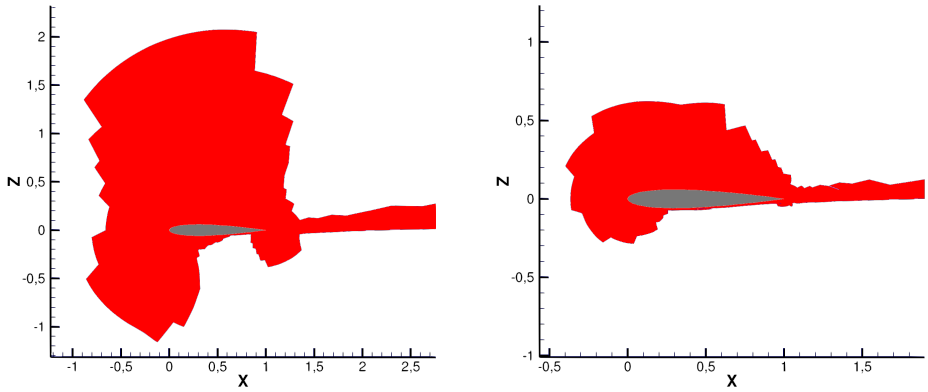
2.2.2 Machine Learning method

The main goal of the ML method here developed is to remove the need of a cut-off input value. Additionally, it is useful to solve all the critical issues of the deterministic methods described in the previous subsection 2.2.1: to exclude the far field cells from the selection, to identify all the element on the body surface and to provide a region selection that is independent of the grid topology.

The attention is focused on *unsupervised* ML methods, to be able to extract the correct identification of the flow regions without any ground-truth label of the results. Two different *clustering* algorithms were tested: the first one is the *k-means*, which is the most simple and widespread algorithm for solving these kind of problems. Good performance are obtained when the distribution of the data in the feature space is such that it is possible to identify different clusters only by considering their mean.

The ML algorithm selected is the *Gaussian Mixture*; all results reported here are indeed obtained by this method. While *k-means* only considers the mean of the data to update the clusters, the *Gaussian Mixture* takes into account the mean as well as the variance of the data. An interesting comparison between the performance of k-means and Gaussian Mixture algorithms is proposed by Patel and Kushwaha in [60] in a completely different context (evaluation of cluster representativeness of the two methods for heterogeneity in resource usage of Cloud workloads). They showed that the k-means, unlike the Gaussian Mixture, was unable in their tests to discover complex non-linear patterns.

The ML method was implemented in a Python code using the *scikit-learn* Python library [71]. The clustering strategy is a *single shot*, without fixing the random seed. However, later in this section, a sensitivity analysis is performed on the random initialization, showing the robustness of the method.



(a) Features: ρ, u,v, p and μ_t. **(b)** Features: ρ, u, v, p, k and ω.

Figure 2.2.3: Failed viscous region selection using primitive variables and unknowns of the turbulence equations. Red: viscous region. NACA 0012, $M_\infty = 0.3$, $Re_\infty = 6 \times 10^6$, $a = 2°$.

Boundary layer and wake.

First two simple test cases are showed: a NACA 0012 airfoil in subsonic ($M_\infty = 0.3, Re_\infty = 6 \times 10^6, a = 2°$) and transonic ($M_\infty = 0.7, Re_\infty = 9.0 \times 10^6, a = 3.25°$) regimes.

In case of subsonic regime, the first attempt of selecting the viscous region was performed just considering the primitive variables (ρ, u, v and p) and the eddy viscosity (μ_t) as input features to the ML algorithm. In Figure 2.2.3a the selection of the viscous region is reported. It is clear that even if the airfoil wake is correctly identified, a large zone of the flow field around the airfoil has been wrongly considered to be boundary layer. Moreover, also elements at the farfield were selected. The results improved by adding, to the primitive variables, the unknowns of the turbulence equations k and ω, as shown in figures 2.2.3b, but a correct selection of the viscous region is not obtained yet.

After several attempts, it was realized that, to obtain a satisfactory clustering, the primitive variable were not sufficient, but the input features have to be discriminatory in identifying the viscous region. A straightforward choice is the adoption of the deterministic sensors $F_{bl_{\mu_t}}$, F_{bl_Φ} and F_{bl_ε} (without any threshold anyway) together with the module of the vorticity $|\omega|$, which is not negligible just in the boundary layer and in the body wake. F_{bl_Φ} revealed redundant, therefore the feature space selected at the end is composed by:

$$F_{bl_{\mu_t}},\ F_{bl_\varepsilon},\ |\omega|\ .$$

The data distribution in the feature space thus obtained is showed in Figure 2.2.4. Each dot represents a grid point of the solution field with the colour identifying the cluster. This is a typical data representation widely used in ML; it allows to visualize the data set in order to have a pre-

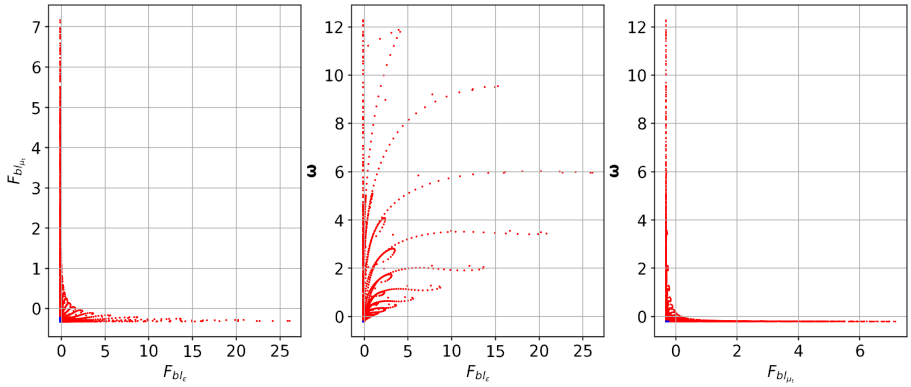
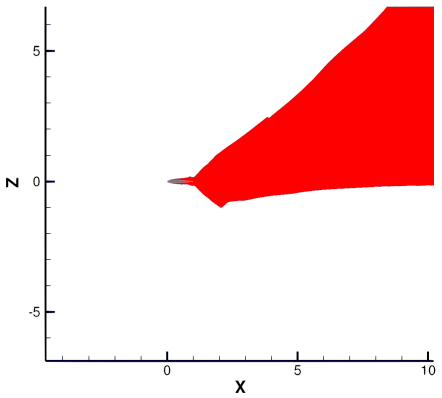


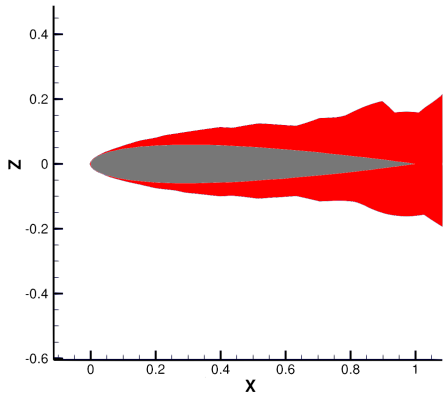
Figure 2.2.4: Data distribution in the feature space using $F_{bl_{\mu_t}}$, F_{bl_ε} and $|\omega|$. NACA 0012, $M_\infty = 0.3$, $Re_\infty = 6 \times 10^6$, $\alpha = 2°$.

liminary idea of which could be the best choice of data, scaling and ML algorithm. A careful analysis of the picture allows to verify if the selected features effectively identify the clusters. In particular, the selected viscous region corresponds to the red points, while the outer region is concentrated in the origin of the axes (in blue). The selected viscous region by the ML algorithm is then compared with the one obtained by the standard deterministic method, and reported in Figure 2.2.5. $F_{bl_{\mu_t}}$ is the sensor adopted in the deterministic method together with topological information to take into account the grid cells in the viscous sub-layer in the laminar part of the flow. The comparison between the two methods clearly shows that the ML algorithm follows the boundary layer and the body wake even better than the deterministic method. The selection by the ML is obtained by just one run, without any input parameter, and it correctly selects the viscous region excluding the far field grid cells, taking into account also the cells on the surface of the airfoil and in the laminar viscous region. On the contrary, the deterministic method required different attempts to obtain the right choice of the cut-off parameter with an additional human activity given by the visual inspection of the selected region. Moreover, with the ML method, there is no need for adding layers of cells around the body surface, therefore, the resulting viscous region is not influenced by the grid topology. Figure 2.2.6 shows the viscous region identified for the same airfoil in transonic regime. The correct performance of the ML algorithm is confirmed. We can notice that the selection of the boundary layer wake is influenced by the vorticity downstream the shock wave which produces an inviscid rotational wake. For this reason the selected region resulted thicker than expected. Hence, in the transonic (and supersonic) case the vorticity is not particularly discriminatory in the selection of the boundary layer wake. This phenomena is also present, and even emphasized, using the deterministic method.

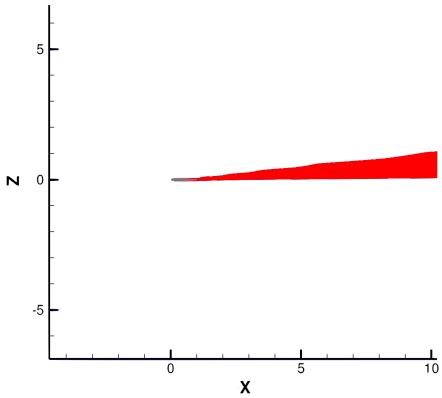
To verify the robustness of the method, to more complicated cases the ML algorithm was tested in case of 3D domains in subsonic and transonic regimes. The subsonic test, in Figure 2.2.7a, consists in a straight wing with $M_\infty = 0.26$, $Re_\infty = 6.6 \times 10^6$ and $\alpha = -1°$. The



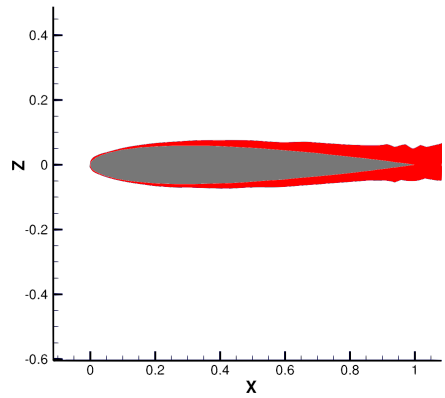
(a) Boundary layer and wake. Deterministic method.



(b) Zoom on the airfoil. Deterministic method.

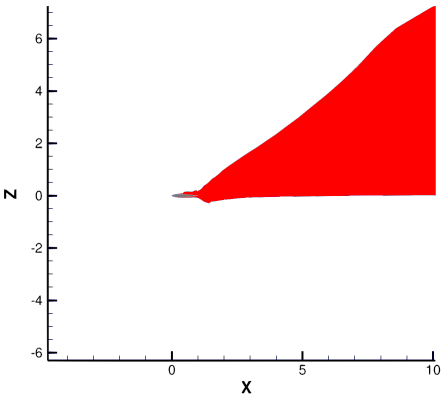


(c) Boundary layer and wake. ML method.

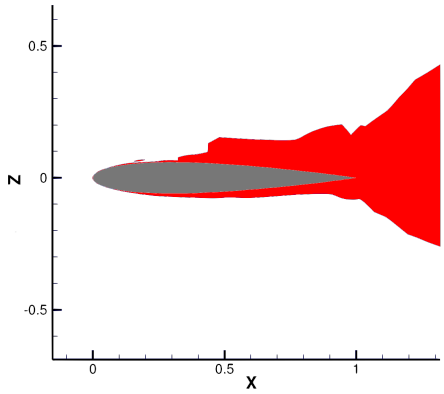


(d) Zoom on the airfoil. ML method.

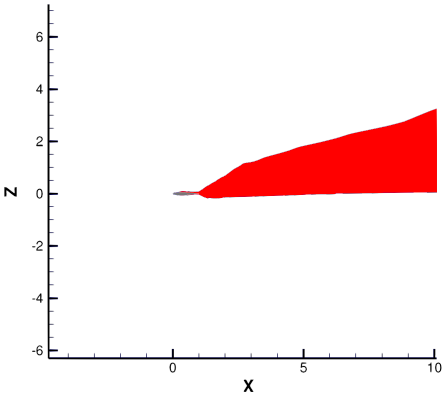
Figure 2.2.5: Selection of the viscous region using the deterministic ($F_{bl_{u_t}}$, $t = 40.0$) and the ML ($F_{bl_{u_t}}$, F_{bl_ε} and $|\omega|$) methods. NACA 0012, $M_\infty = 0.3$, $Re_\infty = 6 \times 10^6$, $\alpha = 2^\circ$. Red: viscous region.



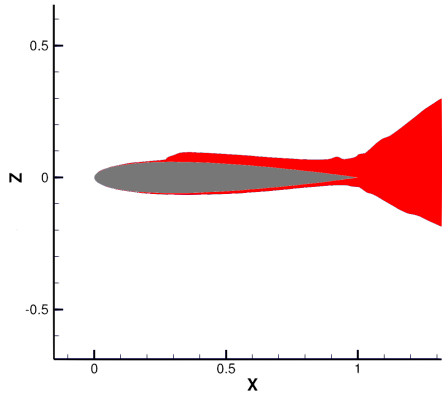
(a) Boundary layer and wake. Deterministic method.



(b) Zoom on the airfoil. Deterministic method.

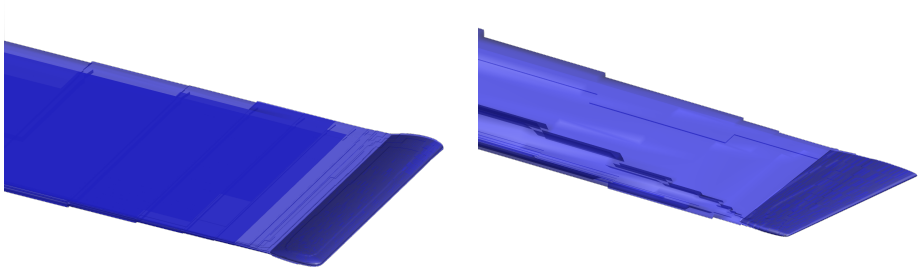


(c) Boundary layer and wake. ML method.



(d) Zoom on the airfoil. ML method.

Figure 2.2.6: Selection of the viscous region using the deterministic ($F_{bl_{\mu_t}}$, $t = 40.0$) and the ML methods. NACA 0012, $M_\infty = 0.7$, $Re_\infty = 9.0 \times 10^6$, $\alpha = 3.25°$, SST turbulence model. Red: viscous region.



(a) Straight wing, $M_\infty = 0.26$, $Re_\infty = 6.6 \times 10^6$, $\alpha = -1°$.

(b) Swept wing (ONERA M6), $M_\infty = 0.84$, $Re_\infty = 11.7 \times 10^6$, $\alpha = 3.06°$.

Figure 2.2.7: Selection of the viscous region for wings in subsonic (a) and transonic (b) regimes using the ML algorithm. Feature space: $F_{bl_{\mu_t}}$, F_{bl_ε} and $|\omega|$. Blue: viscous region.

transonic test, in Figure 2.2.7b, consists in a swept wing (ONERA M6) with $M_\infty = 0.8395$, $Re_\infty = 11.72 \times 10^6$ and $\alpha = 3.06°$. Figure 2.2.7 confirms, also in this case, the good performance of the ML method. The boundary layer selection was obtained with a straightforward adoption of the features $F_{bl_{\mu_t}}$, F_{bl_ε} and $|\omega|$ as in the 2D case. The ML algorithm succeeded also for a more complicated configuration: the NASA Common Research Model wing-body (CRM-WB) configuration† (Figure 2.2.8). In this case, the extension of the wake is limited by the rough resolution of the grid (L1 grid level from the 5^{th} AIAA CFD Drag Prediction Workshop)‡.

Shock wave.

The ML algorithm for the selection of the shock wave was developed independently from the selection of the viscous region. The steps followed to find a suitable feature set are the same as for the selection of the viscous region: a first attempt was done using only the primitive variables and once again the algorithm did not provide a proper region selection.

Again, it is needed to consider variables that are discriminatory for the correct selection of the shock wave region. A preliminary choice is performed by eliminating the regions of the flow in which the formation of the shock wave is theoretically not possible as in the case of expansion zones. Therefore the grid elements where $\nabla p \cdot \frac{V}{|V|} < 0$ were removed from the input of the ML

†https://commonresearchmodel.larc.nasa.gov/
‡https://aiaa-dpw.larc.nasa.gov/Workshop5/workshop5.html

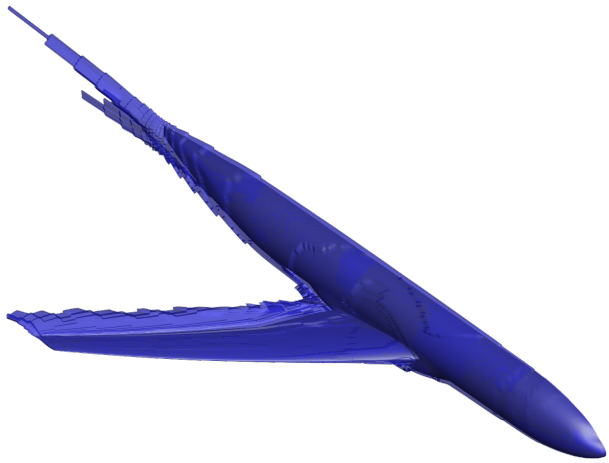


Figure 2.2.8: Selection of the viscous region using the ML method with $F_{bl_{\mu_t}}$, F_{bl_ϵ} and $|\omega|$. NASA CRM wing-body configuration, $M_\infty = 0.85$, $Re_\infty = 5.0 \times 10^6$, $\alpha = 2.30°$, $C_L = 0.510$, SST turbulence model.

algorithm. After several tries, the selected features, referred with **feature-set A**, are:

$$sign(M-1),\ \boldsymbol{\nabla}\rho \cdot \frac{V}{|V|}\,.$$

The features are scaled and translated individually such that the maximal absolute value is 1. The data distribution in the feature space for a 2D transonic case is reported in Figure 2.2.9. In this case, there is a net separation between two groups of data (two clusters). This is an ideal case for the clustering problem, and it is obtained thanks to the combined action of the feature $sign(M-1)$ and the exclusion of the expansion zones. This kind of distribution in the feature space was expected due to the fact that $sign(M-1)$ allows to distinguish data points with $M < 1$ (subsonic) and $M > 1$ (supersonic). Moreover, by neglecting the expansion regions, the algorithm is forced to exclude the data points in the supersonic flow region that cannot belong to the shock wave. In Figure 2.2.9, red and blue dots respectively represent shock wave and outer inviscid zones. The resulting shock wave region for the 2D transonic test case is reported in Figure 2.2.10, where the selection obtained by the deterministic method using the shock sensor reported in equation (2.5) ($t = 0.95$) is also displayed. The deterministic (2.2.10a and 2.2.10b) and ML (2.2.10c and 2.2.10d) algorithms are in agreement; however, the ML method better follows the sonic line in the supersonic region of the flow.

It can be useful to identify different a feature set with which the ML algorithm is able to select grid elements straddling the sonic line. This is referred as **feature-set B** and it is composed by:

$$\text{sign}(M-1),\ \max(0,\ M-1),\ \boldsymbol{\nabla}\rho \cdot V,\ C_p\,.$$

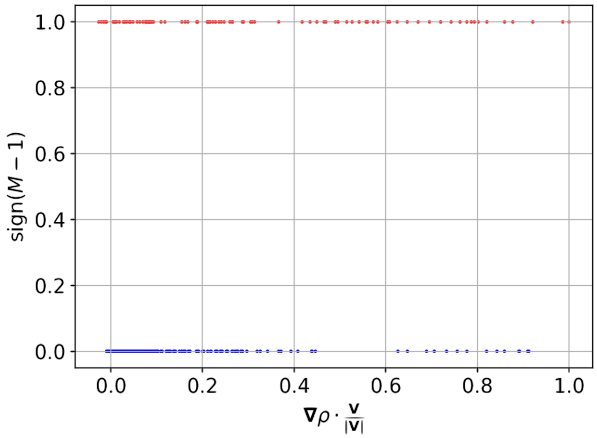
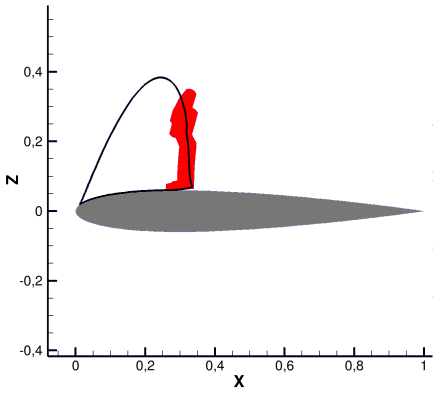


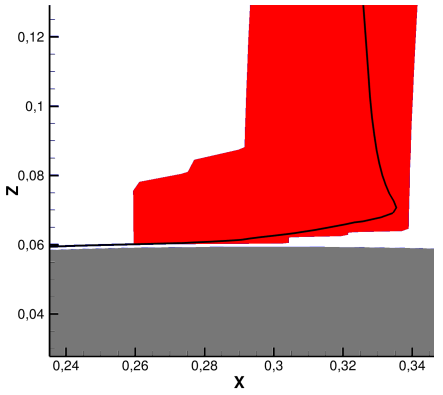
Figure 2.2.9: Data distribution in the feature space using the feature set A: $sign(M-1)$ and $\nabla\rho \cdot \frac{V}{|V|}$. NACA 0012, $M_\infty = 0.7$, $Re_\infty = 9.0 \times 10^6$, $\alpha = 3.25°$, SST turbulence model.

An example of the data distribution of the new feature space is reported in Figure 2.2.11, where the red dots again represent the shock region data points. C_p vs. $\max(0,\ M-1)$ shows that the shock region takes into account also some grid-points where $M \approx 0$, so the foot of the shock almost reaches the body surface. As already discussed for the feature-set A, $\text{sign}(M-1)$ provides a net distinction between subsonic and supersonic data points, despite this, we are able to select also subsonic data points thanks to the introduction of the other features (see for instance C_p vs. $\text{sign}(M-1)$). The same test case reported in Figure 2.2.10 is repeated using the feature set B and shown in Figure 2.2.12. Comparing the foot of the shocks selected using the feature sets A (Figure 2.2.10d) and B (Figure 2.2.12b), the anticipated difference can be identified: in case of B the foot of the shock penetrates deeper in the boundary layer. From the point of view of the drag breakdown, the drag production of a certain amount of elements in the shock-wave/boundary-layer interaction (SWBLI) region will be accounted to the wave drag and not to the viscous drag. This is not a big deal since the drag breakdown in the SWBLI region is ambiguous and there is not a specific rule to partitioning the drag between the two regions. Since the viscous and shock wave regions are identified independently, priority can be assigned to the selection of the boundary layer in the SWBLI region.

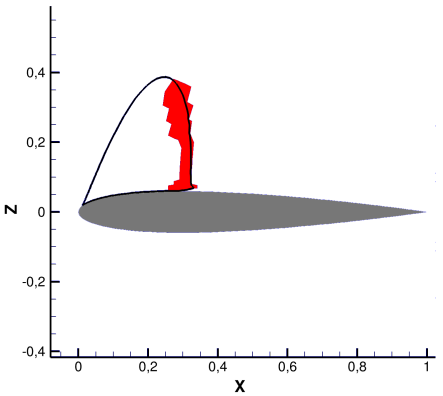
In this case, the ML algorithm was also tested on 3D flows. A successful example for a swept wing (ONERA M6) is reported in Figure 2.2.13. The robustness of the ML algorithm is confirmed by Figure 2.2.14, in case of the NASA CRM-WB configuration. This figure also shows the correct detection of two different shocks, on the wing and on the fuselage canopy, characterized by completely different intensities.



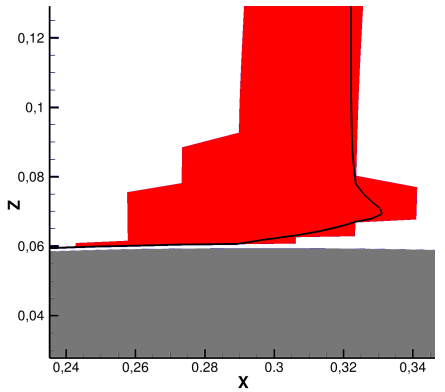
(a) Shock wave region. Deterministic method.



(b) Zoom of the foot of the shock wave. Deterministic method.



(c) Shock wave region. ML method.



(d) Zoom of the foot of the shock wave. ML method.

Figure 2.2.10: Selection of the shock wave region using the deterministic (a), (b) and the ML (feature set A) (c), (d) methods. NACA 0012, $M_\infty = 0.7$, $Re_\infty = 9.0 \times 10^6$, $\alpha = 3.25°$, SST turbulence model. Red: shock wave region. Sonic line in black.

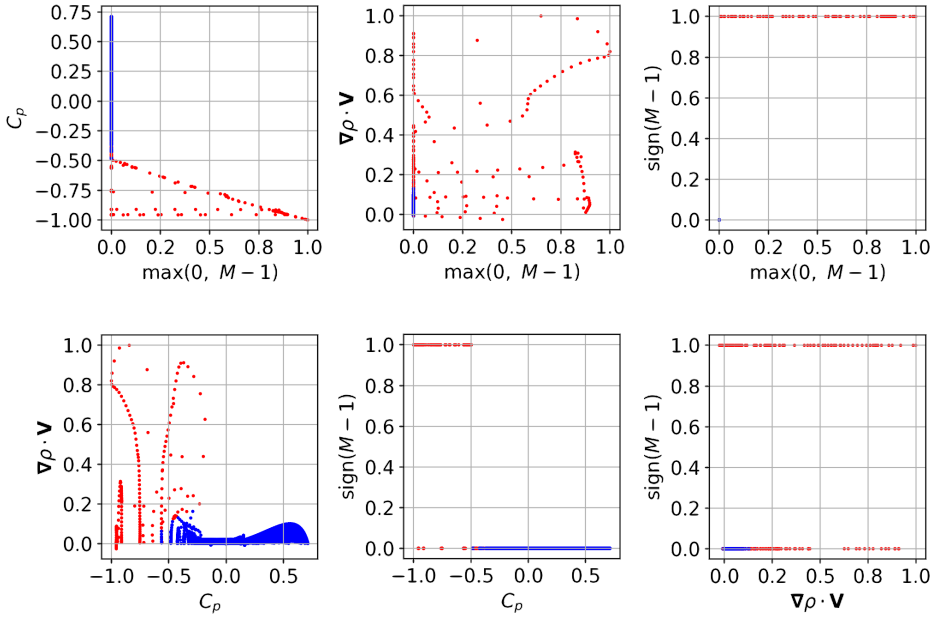
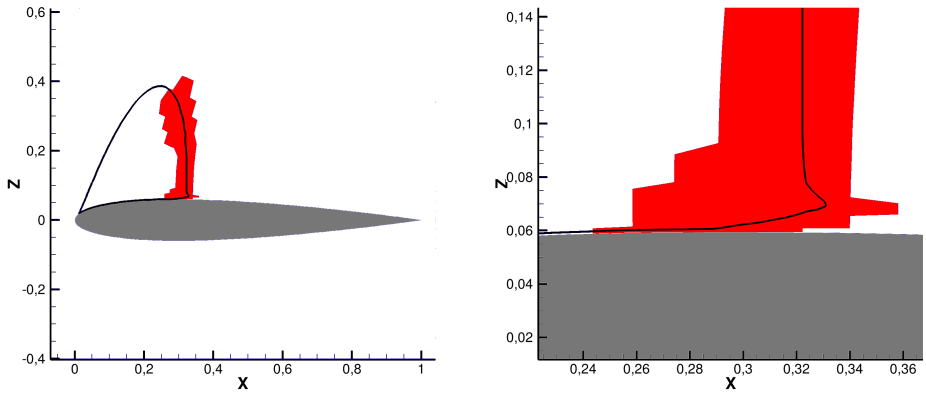


Figure 2.2.11: Data distribution in the feature space using the feature set B: $sign(M-1)$, $max(0,\ M-1)$, $\nabla\rho\cdot V$ and C_p. NACA 0012, $M_\infty = 0.7$, $Re_\infty = 9.0\times10^6$, $\alpha = 3.25°$, SST turbulence model.



(a) Shock wave region. ML method.

(b) Zoom of the foot of the shock wave. ML method.

Figure 2.2.12: Selection of the shock wave region using the ML method with feature set B. NACA 0012, $M_\infty = 0.7$, $Re_\infty = 9.0\times10^6$, $\alpha = 3.25°$, SST turbulence model. Red: shock wave region. Sonic line in black.

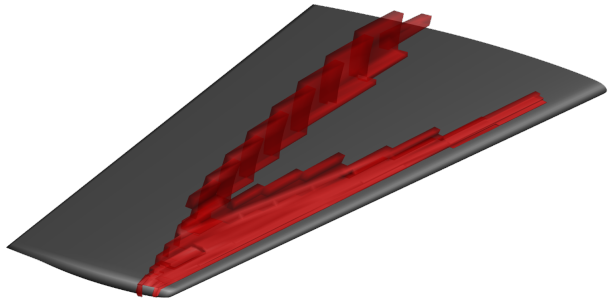


Figure 2.2.13: Selection of the shock wave region using the ML method with feature set B. Swept wing (ONERA M6), $M_\infty = 0.8395$, $Re_\infty = 11.72 \times 10^6$, $\alpha = 3.06°$, SST turbulence model.

Figure 2.2.14: Selection of the shock wave region using the ML method with feature set B. NASA CRM wing-body configuration, $M_\infty = 0.85$, $Re_\infty = 5.0 \times 10^6$, $\alpha = 2.30°$, $C_L = 0.510$, SST turbulence model.

Sensitivity to Gaussian Mixture parameters

Usually, in clustering methods, the number of clusters N is not known a-priori, but a sensitivity analysis on the performance of the clustering algorithm when changing N is needed. In this specific case, the number of regions of interest is already known: boundary layer, shock wave and outer regions. Hence, it was determined to set $N = 2$ when looking for the viscous region in a subsonic flow. However, to assess the convergence and robustness of the adopted Gaussian Mixture algorithm, a sensitivity analysis on the algorithm parameters was performed. A first step consisted in computing the Bayesian Information Criteria (BIC) [72] for different values of N. The analysis is conducted on the NACA 0012 airfoil in transonic regime using the flow features discussed in the previous sections.

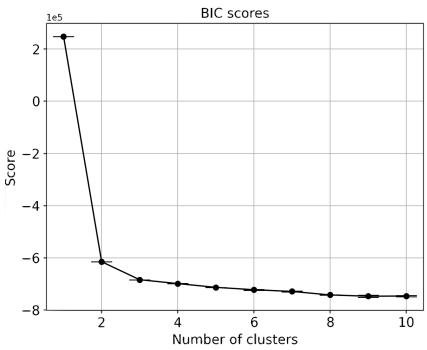
The convergence of the Gaussian Mixture algorithm was already found for $N = 2$ for both the viscous region and the shock wave region identification (Figure 2.2.15). In particular, the BIC score (Figure 2.2.15a, 2.2.15c) is given by

$$\mathrm{BIC} = N\ln(n) - 2\ln(\hat{\mathcal{L}}) \tag{2.7}$$

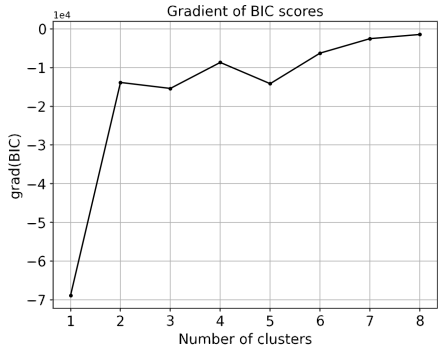
where N is the number of clusters, n is the number of data points and $\hat{\mathcal{L}}$ is the maximum likelihood. Since the objective of the algorithm is to maximize $\hat{\mathcal{L}}$ (as explained in the Appendix), the lower the BIC is, the better the model is. BIC penalizes complex models (with an high value of N) to prevent overfit. However, the gradient of the BIC score (Figure 2.2.15b, 2.2.15d) was employed to evaluate the variation of the BIC score and consequently choose N. A test was also performed to see what happens if $N > 2$ in case of the viscous region selection. In particular, $N = 4$ is considered (Figure 2.2.16). In this case the clusters identified by the Gaussian Mixture are:

- Cluster 0: outer inviscid region.
- Cluster 1: boundary layer and wake.
- Cluster 2: some layers surrounding the airfoil surface.
- Cluster 3: grid points on the airfoil surface.

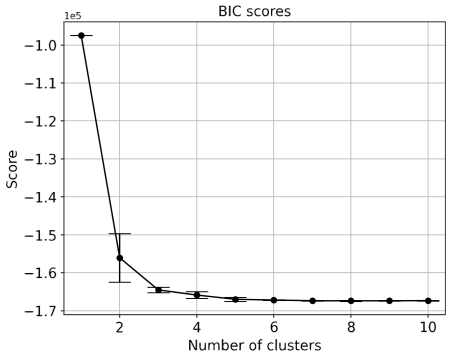
It is interesting to notice that the union of the clusters 1, 2 and 3 found with $N = 4$ perfectly matches the viscous region originally identified with $N = 2$. The selection of the Gaussian Mixture was performed using different initialization methods (k-means, k-means++ and random), the resulting flow-field zones did not change. Finally, the robustness of the selection is assessed by performing 100 initialization of the Gaussian Mixture algorithm for every value of N obtaining the error bars in Figure 2.2.15a (the standard deviations of the BIC scores for $N = 2$ and $N = 3$ are $\mathcal{O}(10^{-5})$), showing a strong robustness of the algorithm on the initialization seed. Additionally, for the case of our interest ($N = 2$), the percentage of the selected viscous region with respect to the entire domain was computed. For the NACA 0012 in subsonic regime, its



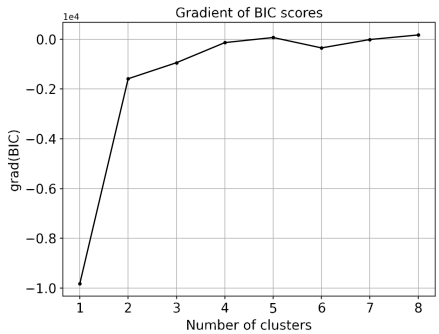
(a) BIC scores for the viscous region



(b) Gradients of the BIC scores for the viscous region



(c) BIC scores for the wave region



(d) Gradients of the BIC scores for the wave region

Figure 2.2.15: Performance of the Gaussian Mixture algorithm changing the number of clusters N. (a), (c) BIC with error bars representing the standard deviation over 100 initialization; (b), (d) gradients of the BIC scores. (a) and (b) are referred to the selection of the viscous region; (c) and (d) to the selection of the shock region.

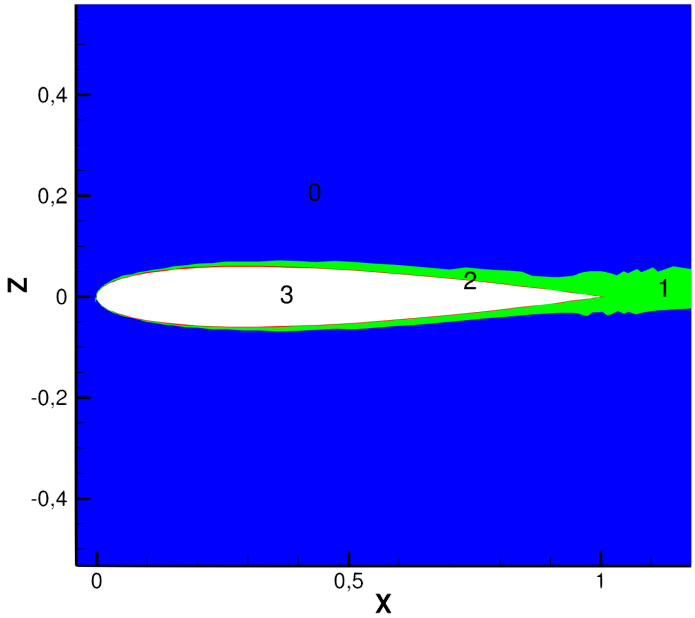


Figure 2.2.16: Selected regions by Gaussian Mixture with $N = 4$ for the NACA 0012 in subsonic regime.

value is always 24.02% over 100 different initialization, proving that the result is independent of the random initialization. The adopted tolerance to stop the EM iterative algorithm is 10^{-10}, with 1000 iteration limit. Best performance are obtained using a *full* or *diag* covariance matrix type (from the *scikit-learn* implementation). The adoption of these parameters for the Gaussian Mixture algorithm leads to a robust region selection for all the cases investigated in this work.

The CPU time to perform a clustering with the Gaussian Mixture on a single core (3.2Ghz clock) is $\sim 0.17s$ with a grid composed by ~ 30000 elements.

Using the Gaussian Mixture model a reliable flow regions decomposition was achieved. At this stage, no comparison was conducted with other ML clustering algorithms, aside from the previously mentioned *k-means*. Consequently, the possibility exists that other clustering algorithms might demonstrate superior performance.

2.3 A practical application: the aerodynamic drag breakdown

The interest in this physical-based domain decomposition appears in different fields: from the optimization of aerodynamic geometries to the aerodynamic forces breakdown. Among the various applications, is here shown the possibility to adopt so-called *far field* methods to decompose the aerodynamic force, drag in particular, in its physical components. These methods are an alternative to classical computations of the aerodynamic force by stress inegration on the body surface, *near field* method. The far field method here adopted is the so-called *thermodynamic*

method, based on the entropy drag concept [68]. The formulas adopted for the drag breakdown are recalled in Appendix B.

The analysis is conducted on three different geometries in transonic regime: NACA 0012 airfoil, RAE 2822 airfoil and NASA CRM wing-body configuration.

2.3.1 NACA 0012

The adopted grids are provided by the NASA website (unstructured 3D hexahedra C-grid). Three levels of grid refinement are considered. The flow conditions for the simulations are $Re_\infty = 9.0 \times 10^6$ and $M_\infty = 0.70$. A grid convergence analysis of the breakdown is reported in Figure 2.3.1, where h (on the horizontal axis) is the normalized averaged grid cell size while the computed drag coefficients are presented on the vertical axis. $h = 4$ is the coarse grid level (64 elements on the airfoil surface), $h = 2$ is the medium grid level (128 elements on the airfoil surface) and $h = 1$ is the fine grid level (256 elements on the airfoil surface). In Figure 2.3.1 dashed

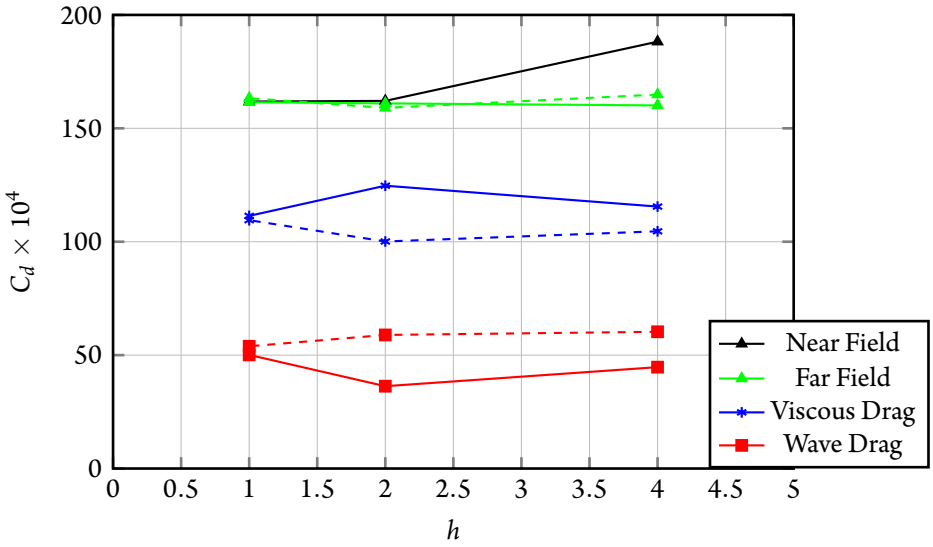
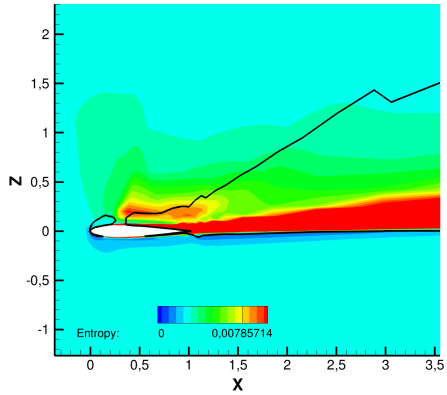


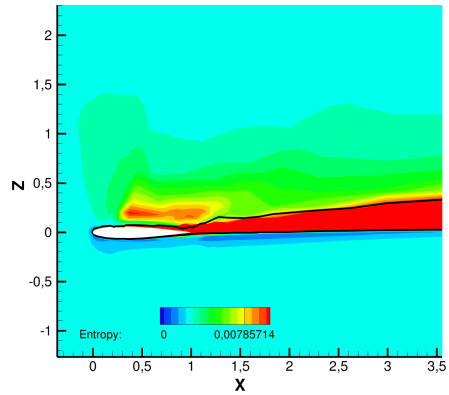
Figure 2.3.1: NACA 0012 grid convergence analysis. SST turbulence model, $Re_\infty = 9.0 \times 10^6$, $M_\infty = 0.70$ and $\alpha = 3.25°$. Dashed line (- - -): Deterministic method; Solid line (—): ML method.

lines are obtained by the deterministic method, while the continuous curves are obtained by the ML algorithm. The green line (—▲—) represent the far field total drag. Comparing it with the near field total drag (—▲—), it is possible to appreciate the effect of the identification of the spurious drag. Indeed the converged total drag value is already obtained on the coarse $h = 4$ grid by the far field method. The deterministic and ML methods are in agreement on the total far field drag.

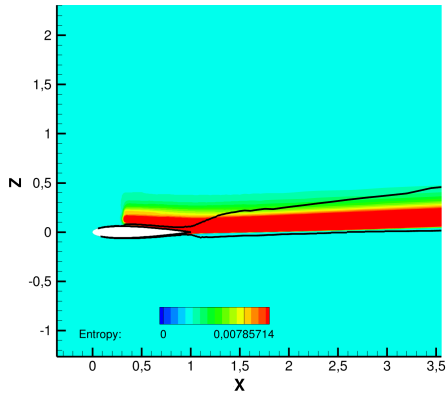
A reliable drag decomposition in physical components is obtained identifying the viscous (—✱—) and the shock wave (—■—) contributions. There is a slight difference between the deterministic and the ML methods in the breakdown on the coarse grids due to a different assignment of the cells in the SWBLI zone. The differences are negligible on the finer grid. Figure 2.3.2



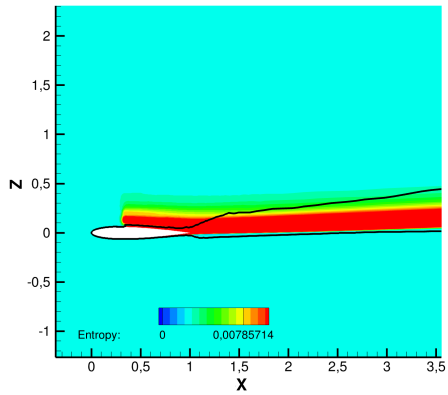
(a) Coarse grid ($h = 4$), deterministic method



(b) Coarse grid ($h = 4$), ML method



(c) Fine grid ($h = 1$), deterministic method



(d) Fine grid ($h = 1$), ML method

Figure 2.3.2: NACA 0012, SST turbulence model, $Re_\infty = 9.0 \times 10^6$, $M_\infty = 0.70$ and $\alpha = 3.25°$. Comparison of the selected viscous region (in black line) by deterministic (sensor $F_{bl_{\mu_t}}$, eq. (2.3)) and ML methods for different levels of grid refinement, contour of the entropy production. (a), (c): deterministic method; (b), (d): ML method.

shows the sensitivity to the grid size of both ML and deterministic selection criteria for the transonic test. The black line identifies the border of the selected viscous zone. The color contour-map represents the computed entropy variation responsible for the local drag generation. The

ML algorithm exhibited independence from the chosen grid, in contrast to the deterministic selection, which was evidently inaccurate in the case of a coarse grid. Figure 2.3.3 shows the lift-

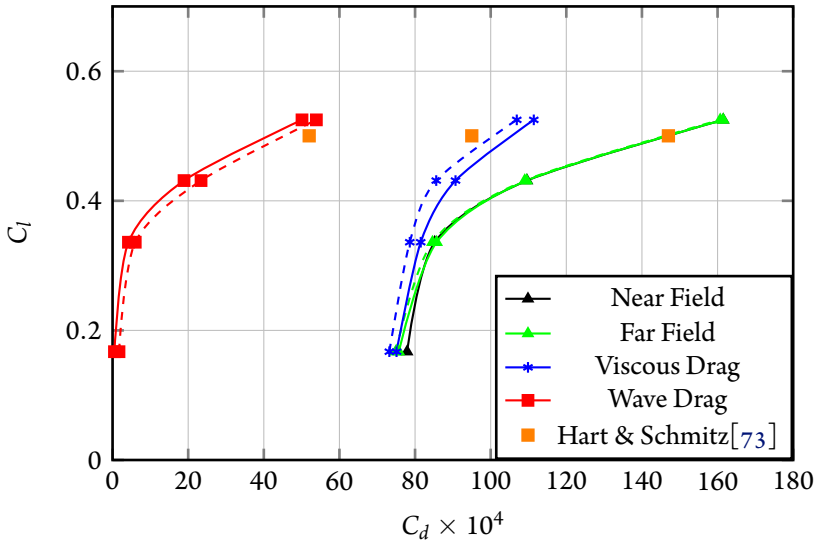


Figure 2.3.3: NACA 0012 fine grid ($h = 1$) lift-drag polar with drag decomposition. SST turbulence model, $Re_\infty = 9.0 \times 10^6$, $M_\infty = 0.70$. Dashed line (- - -): Deterministic method; Solid line (—): ML method.

drag polars computed for the finest grid of the NACA 0012 adopting both ML and deterministic criteria. In the same picture is also proposed the decomposition obtained by Hart and Schmitz [73] (just one point). The results are in substantial agreement, in particular when considering the total drag. The slight differences in the obtained decomposition are due to the already discussed arbitrariness in the definition of the breakdown in the SWBLI region.

2.3.2 RAE2822

A second application is the transonic analysis of the RAE2822 airfoil at $Re_\infty = 6.5 \times 10^6$, $M_\infty = 0.725$ and $\alpha = 2.92°$ ($C_l = 0.796$). Three grid levels are considered: 128, 256 and 512 elements on the surface of the airfoil. Figure 2.3.4 shows the grid convergence analysis and the drag breakdown in viscous (—*—) and wave (—■—) components. Again a converged drag decomposition was already obtained with the coarse grid level. Refining the grid, the differences in the selected regions are hardly visible; we just report the selection for the coarse grid, Figure 2.3.5. Present results are compared with those obtained by Hart and Schmitz [73]. Free-stream conditions adopted by Hart and Schmitz ($M_\infty = 0.73$ and $\alpha = 2.72°$) are slightly different; however, the agreement in the computed drag breakdown is quite satisfactory.

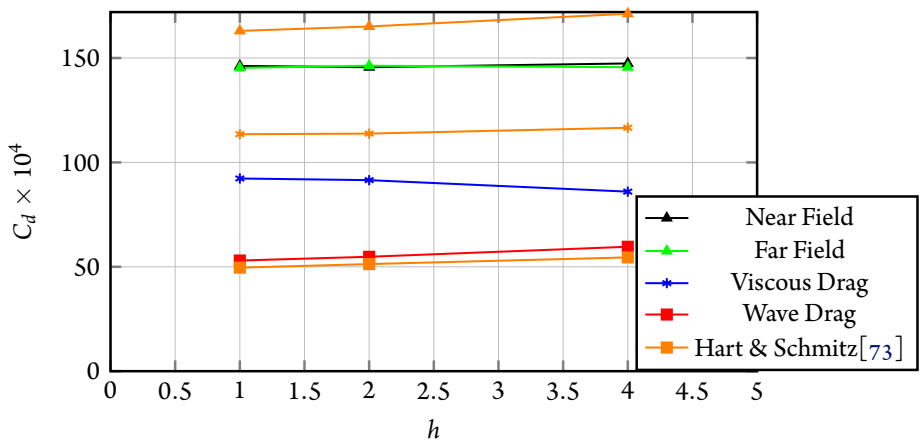
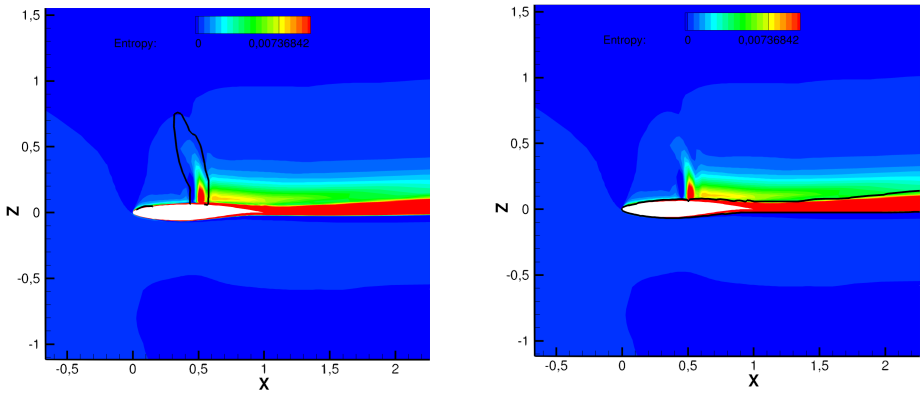


Figure 2.3.4: RAE2822 grid convergence analysis, SST turbulence model, $Re_\infty = 6.5 \times 10^6$, $M_\infty = 0.725$ and $\alpha = 2.92°$.



(a) Shock wave region **(b)** Viscous region

Figure 2.3.5: RAE2822 coarse grid ($h = 4$), SST turbulence model, $Re_\infty = 6.5 \times 10^6$, $M_\infty = 0.725$ and $\alpha = 2.92°$. Selection of the regions (in black line), contour of the entropy production.

2.3.3 NASA CRM WING-BODY CONFIGURATION

The ML algorithm was also tested in a more complex 3D case: the NASA CRM-WB configuration. The mesh is available in the website of the 5th AIAA CFD Drag Prediction Workshop. It is the coarsest grid level (L1) composed by just 638,976 hexahedral elements [74]. Following the CFD analysis by SU2, the ML algorithm was able to select the viscous and the shock regions to be provided to the drag breakdown method; Figure 2.2.8 and 2.2.14 show the selected viscous and shock wave regions for a particular angle of attack. Figure 2.3.6 shows the lift-drag polar with the obtained drag breakdown in lift-induced, viscous and wave drag components based on the ML region selection. The black line —▲— is the near-field total drag; after removal of the spurious drag (thanks to the region selection), the computed total drag was corrected in the far-field drag (green line —▲—). The results proposed by Scalafani et al. [75] at the 5th AIAA CFD Drag Prediction Workshop on the same grid are in agreement with present ones and therefore are not presented. However the authors performed a grid convergence analysis just at $C_L = 0.5$. The magenta asterisk (*) is the near field value they obtained on a very fine grid built-up by 43.3 million elements. Assuming the computational cost proportional to the number of grid elements, the identification of the spurious drag by the drag breakdown method allowed for the same accuracy reducing the computational cost of a factor 70.

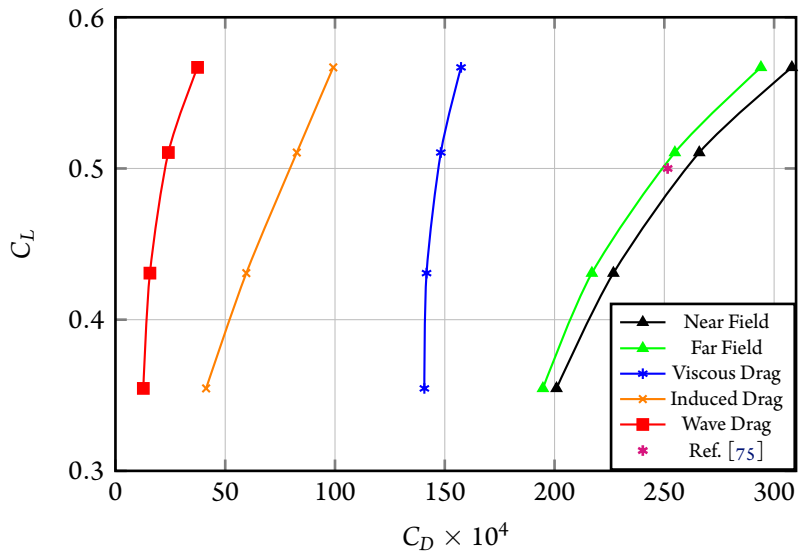


Figure 2.3.6: Lift-drag polars of the NASA CRM-WB configuration at $M_\infty = 0.85$, $Re_\infty = 5.0 \times 10^6$, k- SST turbulence model.

Also in this case, it is important to notice that the ML algorithm successfully identified the regions for all the angles of attack used for the drag breakdown analysis without any user interaction.

The results have been also compared with those obtained by other recent studies, in particular by Hart et al. [76] and by Fournis et al. [77]. Table 2.3.1 summarize the decompositions at the design lift coefficient $C_L = 0.5$. Taking into account that the three compared methods are different and despite the adoption of a much coarser grid, present results are in substantial agreement with the recent literature data.

	C_D	C_{D_i}	C_{D_v}	C_{D_w}
Present Study	254.8	82.6	148.2	24.1
Hart et al. [76]	257.0	87.4	148.8	20.9
Fournis et al. [77]	254.7	87.3	155.1	12.5

Table 2.3.1: Drag decomposition of the NASA CRM-WB configuration at $M_\infty = 0.85$, $Re_\infty = 5.0 \times 10^6$ and $C_L = 0.5$. Comparison with literature predictions.

These results, puts light on the capabilities of data-driven methodologies in the field of the CFD, by providing a powerful tool for the post-processing of CFD solutions. Indeed, present algorithm can now be embedded within iterative loops for optimization problems, not a feasible task by classical deterministic methods. The clustering algorithm was here applied to RANS solutions, but it can also be adopted for the analysis of LES and DNS properly averaged data.

3

Aerodynamic Analysis by Autoencoders

In this chapter, the investigation centers on assessing the effectiveness of an unsupervised ML strategy, specifically *convolutional autoencoders* (AE), in predicting the aerodynamic characteristics of classical wing cross-sections at high Reynolds numbers. The primary objective is to illustrate the capability of ML to accurately predict flow fields around airfoils with diverse geometries and operating conditions in real-time. Furthermore, the aim is to demonstrate the AE's proficiency in generating solutions for airfoils not encompassed in the original training set, effectively covering a continuous range of wing sections.

The significance of quasi real-time flow prediction is underscored in various domains of Fluid Mechanics, encompassing applications such as active flow control and the acceleration of shape optimization algorithms. Employing AEs as generative models holds the potential to fulfill this objective, thereby catalyzing the advancement of novel solutions in aerodynamics. As anticipated in Section 1.2.3, AEs are already employed in the field of Fluid Mechanics. However, the approach undertaken in this work is designed to address existing limitations and explore unanswered questions within the field. Throughout the chapter, the novelty of this work is emphasized concerning current solutions, as detailed in section 1.3.

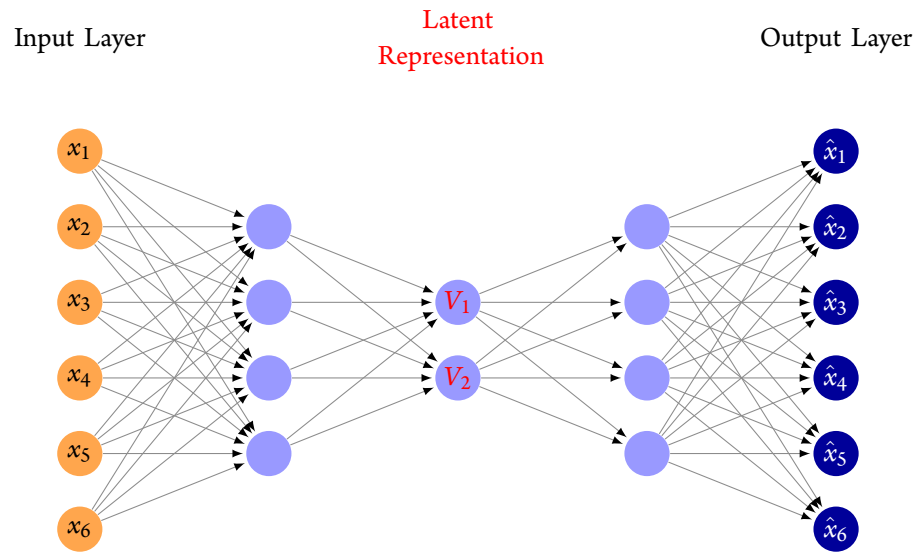


Figure 3.1.1: Schematic representation of a fully connected autoencoder.

3.1 An introduction to autoencoders

AEs are unsupervised deep-learning algorithms, which fundamentally target data compression, i.e. reducing the dimensionality of the input data [22]. The architecture of an AE consists of two main elements, the encoder and the decoder. Giving an input dataset, AEs construct (*learn*) an equivalent low dimensional representation, termed the *latent space* using the *encoder* portion of the network. On the other hand, the *decoder* rebuilds the input from the latent space.

The functioning of a fully connected AE is similar to that of the NN illustrated in section 1.1.1, with differences in the NN **architecture** and fundamental **purpose**. A straightforward illustration, depicted in Figure 3.1.1, can facilitate understanding of the AE's operation. In this specific instance, the AE processes a six-element vector $\mathbf{x}$ as input and produces, in output, a reconstruction of the input, denoted by $\hat{\mathbf{x}}$. The input is propagated through the NN, computing the outputs of each intermediate layers as in eq.(1.4) (same procedure discussed in section 1.1.1).

The primary distinction from a simple NN, lies in the identification of two sections in the architecture of an AE. The first segment comes before the *latent representation* and goes by the name of *encoder*. The second follows the *latent representation* and is known as the *decoder*. It is worth noting that in this instance, the encoder and decoder consist of just one hidden layer for the sake of simplicity. The two portions of the NN have distinct roles:

- **Encoder**: to perform data compression and feature extraction. This is accomplished by reducing the number of neurons in the hidden layers until the latent space is attained. In this specific example, two latent variables (V_1 and V_2) are extracted from the input. As

a result, the high-dimensional input is compressed in a low-dimensional representation: $\mathbb{R}^6 \rightarrow \mathbb{R}^2$.

- **Decoder**: to reconstruct the input data. This is accomplished by taking in input the latent variables, and gradually increasing the number of neurons in the hidden layers until the output layer is reached and the data-dimensionality is recovered: $\mathbb{R}^2 \rightarrow \mathbb{R}^6$.

The AE is typically trained in an unsupervised manner: the input and the output are the same. Thus, the objective function during the training phase is to minimize the error between the input and its reconstruction in output. The latent representation is "chosen" by the AE and is determined by extracting the most essential features required to describe the input dataset accurately (more complex loss functions can be employed, depending on the specific case, see for instance eq.(3.2)). As previously detailed in section 1.2.3, this technique can be considered as a non-linear variation of classical POD, where the latent variables serve as equivalents to the POD modes [28, 48, 78].

The encoding-decoding procedure can be summarized mathematically as:

$$\mathbf{V} = e(\mathbf{x}); \qquad \hat{\mathbf{x}} = d(\mathbf{V}); \qquad \hat{\mathbf{x}} = f(\mathbf{x}) = d(e(\mathbf{x})) \ . \tag{3.1}$$

where $e()$ is the *encoder* function, $d()$ is the *decoder* function, $f()$ is the entire transformation *encoder+decoder*, $\mathbf{V}$ is the vector of latent variables (also referred to as *latent space*) and $\mathbf{x}$ and $\hat{\mathbf{x}}$ are respectively the input and its reconstruction.

In this dissertation, deep-AEs are adopted, and one or more convolutional layers are employed in both the encoder and the decoder, leading to the networks under investigation being referred to as Convolutional AE. A scheme of a deep Convolutional AE is reported in Figure 3.1.2, where the input and output of the AE are tensors:

$$\mathbf{x}, \ \hat{\mathbf{x}} \in \mathbb{R}^{n \times m \times c}; \qquad \mathbf{V} \in \mathbb{R}^{N_{ls}} \ .$$

where $n \times m \times c$ is the dimension of the input dataset and N_{ls} is the dimension of the *latent space* (the most important hyperparameter for an AE). In this illustrative Figure, there are 3 convolutions and 2 fully connected layers for the encoder (the decoder is mirrored).

AEs are defined in terms of N_e and N_d *free* parameters in the encoder and decoder respectively. These parameters (ℓ_e and ℓ_d) are obtained via an optimization algorithms aimed at minimizing the difference between the input and the reconstructed data, i.e. $x - \hat{x}$ (details regarding the training phase are deferred to section 1.1.1). In addition, the cost function for the optimization (the loss) can also includes regularization terms that aims at reducing overfitting or physical constraints.

In the present work the loss function is a combination of the reconstruction error by the mean squared error (MSE) to reduce the reconstruction error and a regularization term applied to the learnable parameters of the AE. Typically, this regularization term involves a penalty on

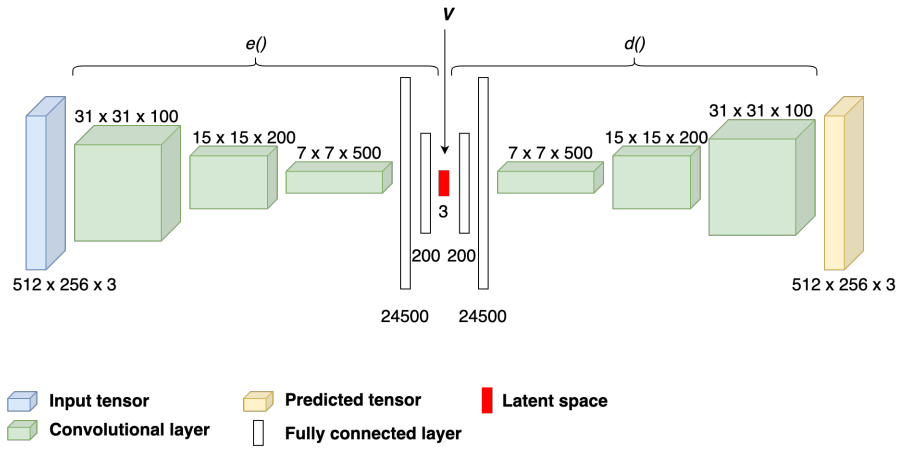


Figure 3.1.2: Schematic representation of the adopted convolutional autoencoder.

the L_2 norm:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 + \lambda\frac{1}{N_{ls}}\sqrt{\sum_{j=1}^{N_{ls}} V_j^2} \tag{3.2}$$

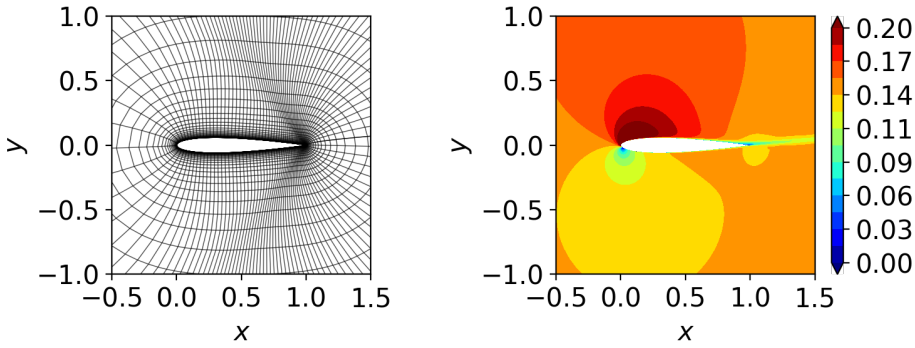
where, the first term is the MSE between the input and its reconstruction, while the second term is the regularization term. N is the number of data, V_j is the j-th latent variable and N_{ls} is the latent space dimension. λ is a weight of the regularization term. In particular, in section 3.3, an analysis of the effect of the kind of regularization on the latent representation of the AE is presented. Additionally, the weight of the regularization term also have an interesting impact on the latent representation as demonstrated in subsection 3.5.2. In all the cases presented in this dissertation, training is carried out using the *Adam* optimizer [79], and *batch training* is employed. A rectified linear unit ($ReLU$) serves as the activation function after each layer, except for the output layer, where an *Identity* function is used, as the primary focus is on regression tasks. Specific AE architectures and hyperparameter choices for each numerical experiment are detailed within their respective sections. It's important to note that the baseline model, which serves as a reference, is illustrated in Figure 3.1.2, with minor adaptations made for different applications. For a more in-depth exploration of hyperparameter and architecture selection, please refer to chapter 4.

All the results presented in this dissertation were obtained developing *Python* code with the implementation of AEs carried out through the *PyTorch* libraries [80].

3.2 Database definition

A critical element of data driven models is the dataset preparation for the training of the ML algorithm. Figure 3.2.1 shows an example of a mesh around a NACA 0012 airfoil adopted for

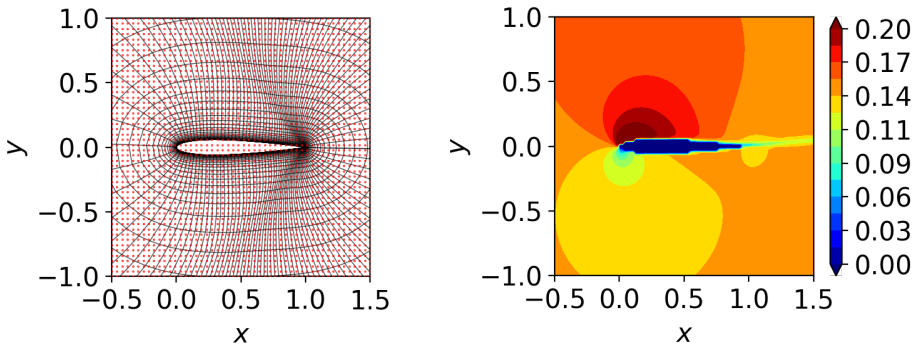
a RANS simulation. It is an O-grid with 512 elements on the airfoil surface and 256 in normal direction. The farfield is located at 50 chords far from the airfoil. A first difficulty arises from the



(a) Mesh **(b)** Local Mach number

Figure 3.2.1: (a) CFD grid around a NACA 0012 airfoil; (b) local Mach number contour, SU2 RANS simulation: $M_\infty = 0.15$, $Re_\infty = 5.0 \times 10^6$, $a = 8.0°$, SA turbulence model.

mesh topology: it is not a cartesian grid and it presents refinements towards the airfoil surface. This limitation is a well-known aspect of such algorithms, as CNNs require a matrix as input. There are more sophisticated architectures documented in the literature, but these are beyond the scope of the current focus. A first approach widely adopted in literature [45, 49, 81, 82] is to construct a cartesian grid where to interpolate data from the CFD mesh. An example of this approach is showed in Figure 3.2.2, where the red dots represent the cartesian grid where data are interpolated. The Mach number contour in Figure 3.2.2 is obtained with 10000 points in the



(a) Mesh **(b)** Local Mach number

Figure 3.2.2: (a) CFD grid around a NACA 0012 airfoil, red dots: overlayed cartesian grid; (b) local Mach number contour interpolated on the cartesian grid, SU2 RANS simulation: $M_\infty = 0.15$, $Re_\infty = 5.0 \times 10^6$, $a = 8.0°$, SA turbulence model.

interval $x \in [-0.5,\ 1.5]$ and $y \in [-1.0,\ 1.0]$. Clearly, the accuracy on the airfoil surface is lost, and there is no possibility to extract surface quantities. To increase the accuracy, the overlayed cartesian grid must be refined and composed by much more elements than the original CFD mesh. This would dramatically increase the computational cost for the ML analysis, and at the same time, it is not possible to recover the surface data.

To overcome these limitations, an *i-j* mapping of the computational grid is proposed. An example is reported in Figure 3.2.3. The *i-j* mapping permits a cartesian representation of the CFD

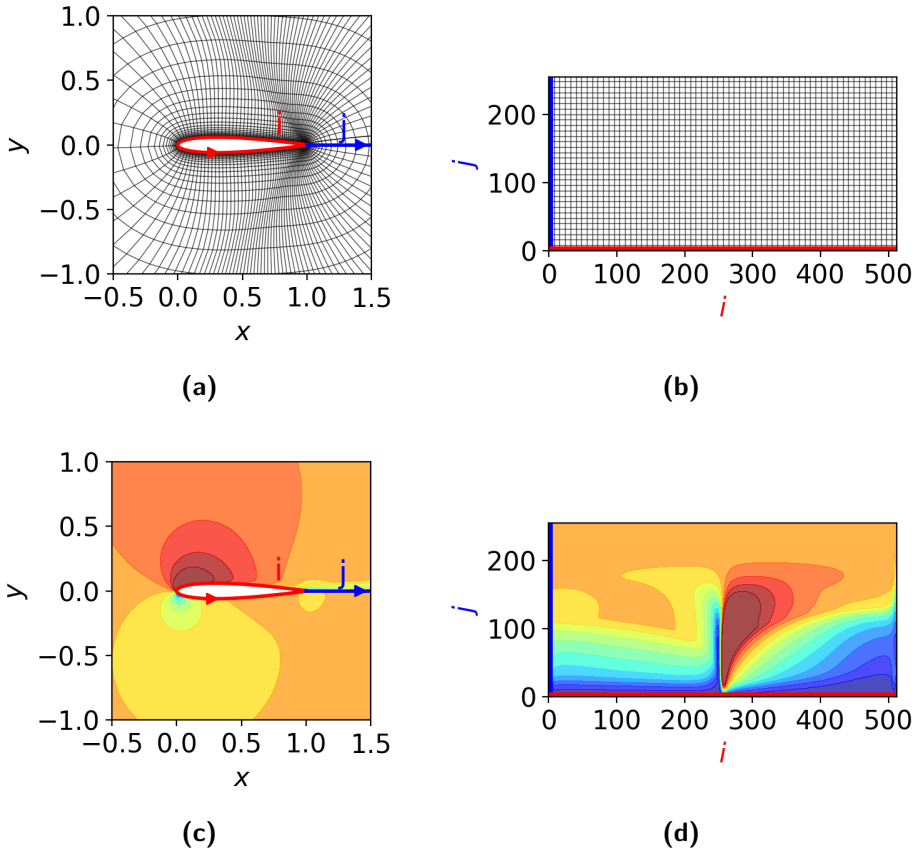


Figure 3.2.3: Transformation of the grid and the corresponding RANS solution into the *i-j* mapping of the computational grid. (a) computational grid in the physical plane (*x-y*); (b) computational grid in the *i-j* plane; (c) local Mach number contour in the physical space; (d) local Mach number contour in the *i-j* plane.

solution, while keeping the same accuracy of the CFD solution. The field variables are organized in matrices using the *i* and *j* indexes of the mesh. The *i* index is a curvilinear coordinate on the airfoil surface and it corresponds to the horizontal axis of the encoded mapping, while the *j* index tracks the radial direction away from the airfoil and corresponds to the vertical axis encoded

mapping. This input representation has also the advantage of amplifying the boundary layer as j does not follow the mesh clustering at the airfoil surface (see Figure 3.2.3 (d)); this in turns enables to highlight the boundary layers in the training step of the ML algorithm.

Adopting this mapping, the geometrical information are lost. To deal with this problem, the geometry needs to be fed into the neural network in some way. The most simple and direct approach is to consider the x and y coordinates in the physical space. The input to the AE is composed of a tensor, where the first two dimensions are determined by the i-j maps, and the third dimension represents the channels. This corresponds to the number of i-j maps provided as input, i.e., the input variables.

All the results presented in this work are obtained with datasets constructed with an i-j mapping of the CFD solution.

3.2.1 RGB images vs physical properties

In the ML framework, with a wider perspective, CNNs are employed for image processing and computer vision applications, primarily working with RGB or gray-scale images. When dealing with RGB (red-green-blue) images, the construction of the input tensors to the convolutional neural network is trivial: each case will be a tensor which dimensions are the x and y pixel dimensions and 3 channels corresponding to the R, G and B colors. For aerodynamic problems, the determination of the appropriate data structure is not so simple. Consequently, it's essential to explore the effectiveness of using RGB images to adequately represent physical solutions in the context of flow fields.

The investigation was performed on two aerodynamic datasets:

DS1: linear case (33 RANS solutions), changing 2 parameters:

- airfoil curvature (NACA 2412, 4412, 6412),
- angle of attack $\alpha \in [-5°,\ 5°]$.

DS2: non-linear case (124 RANS solutions), changing 2 parameters:

- airfoil maximum thickness t (NACA 0012, 0014, 0016 and 0018),
- angle of attack $\alpha \in [0°,\ 30°]$.

The input to the convolutional AE consists of 2D matrices (i-j mappings) each with 5 channels (pressure coefficient C_p, vorticity ω, local Mach number M, x and y spatial coordinates), and the variables are scaled between 0 and 1. The datasets are composed of either images (RGB channels) or raw solutions (matrix of floating points). Therefore, the number of input channels is 15 for the images (3 RGB $\times$ 5 variables) and 5 for the raw data (1 channel $\times$ 5 variables). In DS1 the aerodynamic behaviour is linear, while in DS2 it is non-linear since the airfoil stall occurs. The investigation is aimed to evaluate the embedded (latent) representation learned by the AE. The AE architecture is the same reported in the schematic representation in Figure 3.1.2, using a

latent space dimension $N_{ls} = 3$. The latent space dimension is the most important hyperparameter of an AE. By increasing N_{ls} it is possible to reach a certain level of tolerance in the training process with less epochs. However, a high dimensional latent space can lead to overfitting and ultimately, poor performance on new unseen cases. In this work, another critical design goal is interpretability: a very high dimensional latent space limits the ability to extract physical understanding from the latent variables. In this specific case (DS1 and DS2), 2 parameters are varying in the database: the airfoil thickness and the angle of attack. For this reason, at least 2 latent variables are necessary, to describe the dataset. It is also possible to see this from Figure 3.2.4, where the training history of the AE is reported. Clearly, $N_{ls} = 1$ is not sufficient for reaching a converged training. On the other side, $N_{ls} = 2$ and 3 lead to convergence.

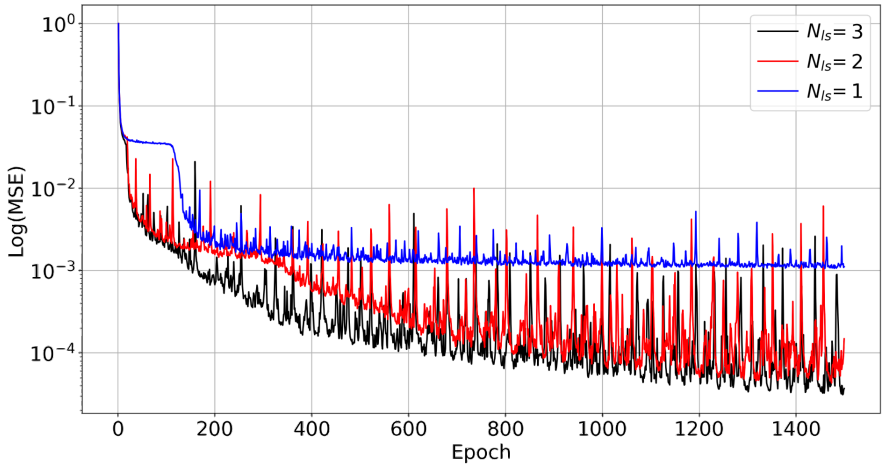
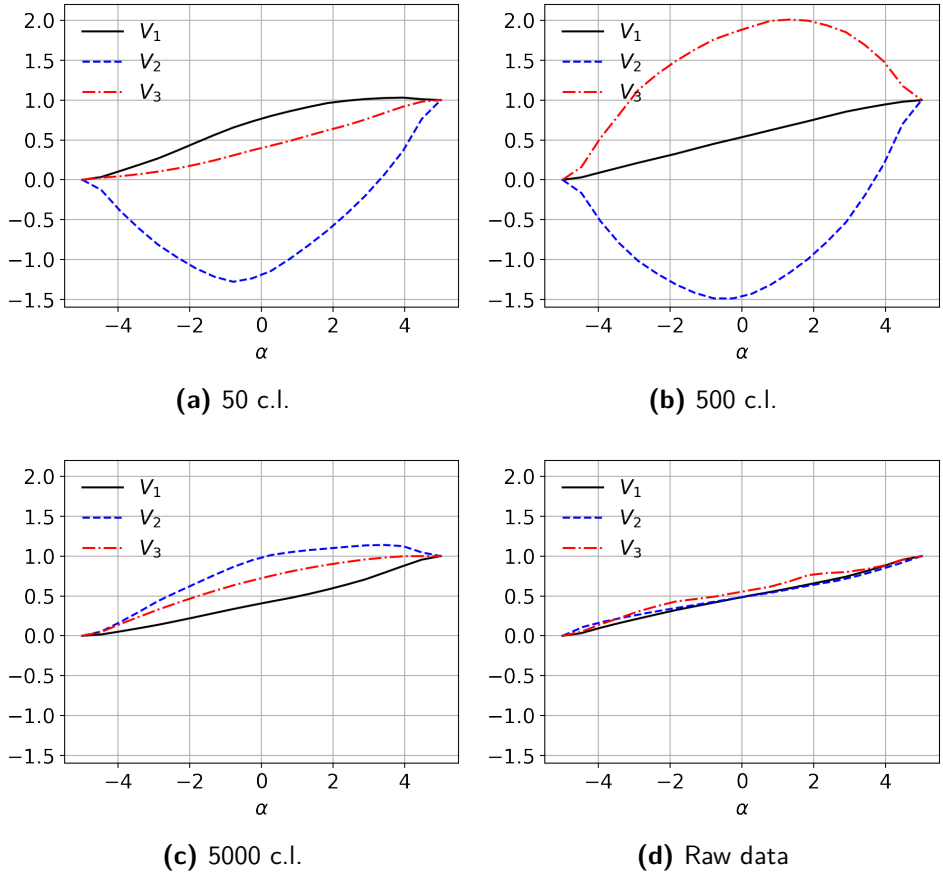


Figure 3.2.4: Convergence history of the training process with DS1. Comparison of different latent space dimensions: $N_{ls} = 1, 2, 3$.

Figure 3.2.5 shows normalized latent variables learned by the AE trained with images constructed using different numbers of contour levels: 50, 500 and 5000 respectively. These are also compared to using directly the raw data. It is clear that the latent variables reported as a function of the angle of attack are quite sensitive to the input format. It is expected that in these conditions the latent space represents closely the variability expressed by the angle-of-attack. The results confirm that increasing the number of contour levels in the image (e.g. increasing the precision in representing the data) leads to a better correlation between the latent variables and the expected variability. Using the raw data as input leads to a collapse of the latent space and effectively a purely linear response to the changes in the angle-of-attack. Figure 3.2.6 shows the same sensitivity analysis carried out in the non-linear regime. As before, there is a sensitivity of the latent space to the number of contour levels. Moreover, the raw data in input produced latent variables which are linear for low angle-of-attack consistently with what shown previously. The latent representation becomes non-linear when stall occurs. For a better visualization of the latent variables using raw data, they are reported in Figure 3.2.7 reducing the y-axis scale. RGB



(a) 50 c.l. **(b)** 500 c.l.

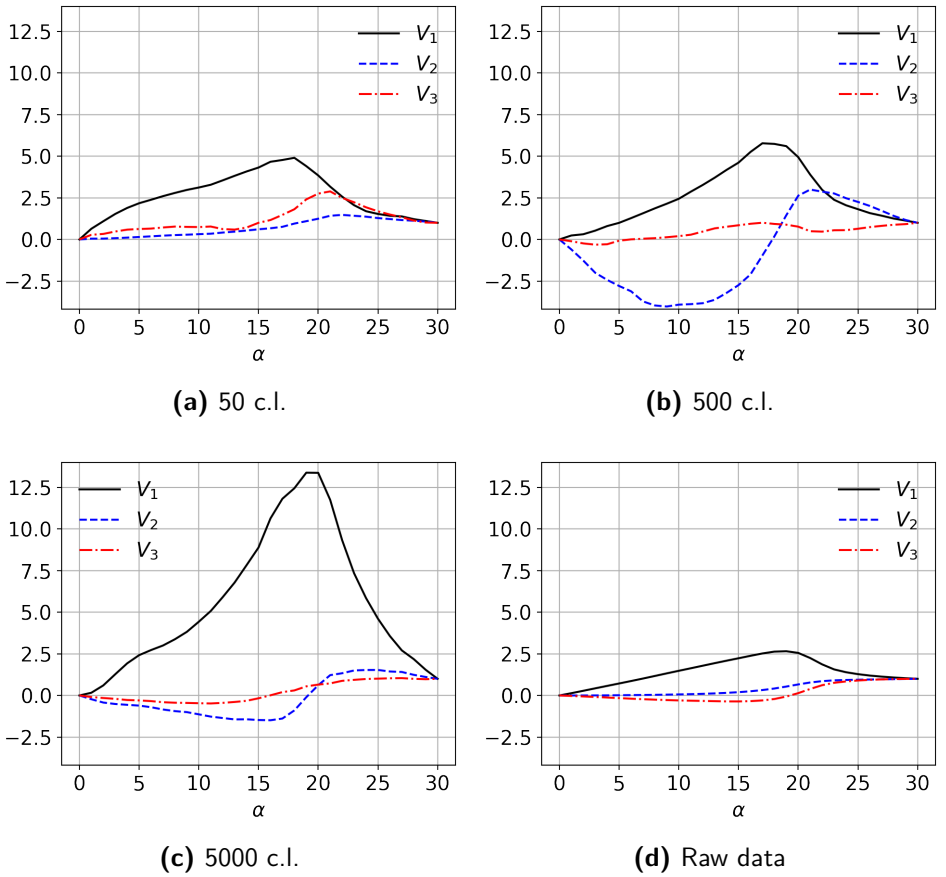
(c) 5000 c.l. **(d)** Raw data

Figure 3.2.5: Latent variables extracted by the AE (normalized between 0 and 1). Linear case: NACA 6412. Comparison between raw data and RGB images with different number of contour levels (c.l.). Analysis performed on dataset DS1.

images introduce a loss of information which depends on the number of contour levels adopted to build the image. In particular, using raw data, each element of the input tensor will be a real number, while, the accuracy of the RGB images depends on the contour bands present in the image. When raw data are adopted, the latent representation learned by the AE reflects the physical behaviour of the phenomenon. Downstream of this analysis all the datasets generated in this work use raw data.

3.3 Regularization

As described by Newson et al. in [83], different regularization methods can lead to different behavior of the latent variables and different reconstruction performance by the AE. The sen-



(a) 50 c.l. **(b)** 500 c.l.

(c) 5000 c.l. **(d)** Raw data

Figure 3.2.6: Latent variables extracted by the AE (normalized between 0 and 1). Non-linear case: NACA 0014. Comparison between raw data and RGB images with different number of contour levels (c.l.). Analysis performed on dataset DS1.

sitivity of the present algorithm to the use of L_2 norm regularization on the encoder and/or decoder weights and on the latent variables was investigated. Eventually it was found that the L_2 norm regularization applied only to the encoder weights leads to a tidy and compact latent space which is amenable to interpretation. As a comparison of the various types of regularization, the latent variables obtained using the non-linear regime dataset and $N_{ls} = 3$ are presented in Figure 3.3.1: the results are not vastly different, although it is clear that regularizing the encoder weights leads to a compact latent space. The latent space regularization in Figure 3.3.1b is obtained by applying a penalization on the L_2 norm of the latent variables, while the encoder weights regularization in Figure 3.3.1c is an L_2 norm of the weights of the Encoder.

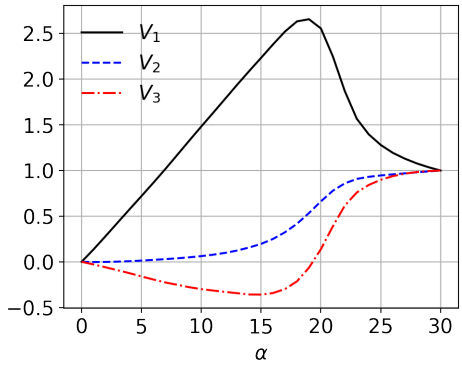
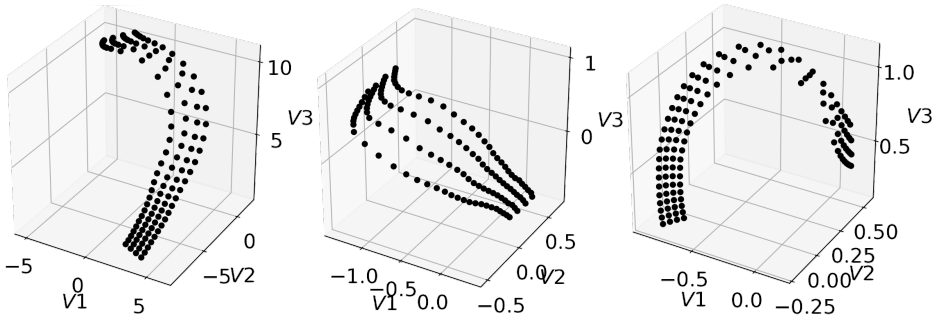


Figure 3.2.7: Latent variables extracted by the AE (normalized between 0 and 1) using raw data in input. Non-linear case: NACA 0014 (same result reported in Figure 3.2.6(d), but with a different y-axis scale).



(a) No regularization **(b)** Latent space regularization **(c)** Encoder weights regularization

Figure 3.3.1: Comparison of latent spaces for $N_{ls} = 3$ changing the regularization technique. AE trained with the non-linear data-set.

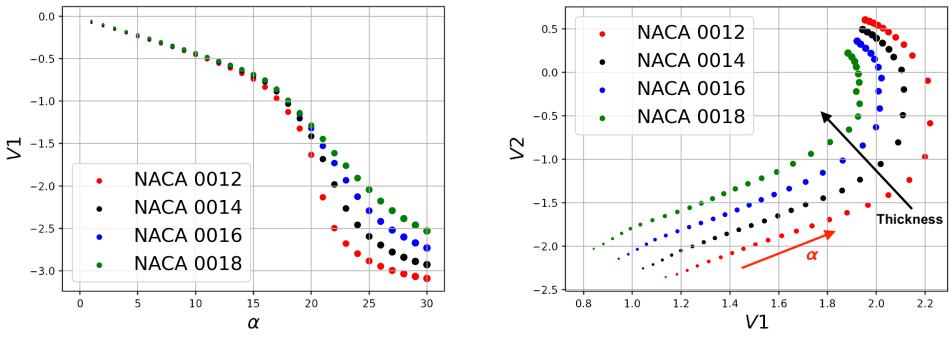
3.4 Latent space investigation

Based on the sensitivity analysis illustrated earlier, the AE was trained applying the L_2 norm regularization on the encoder weights. The non-linear regime dataset (DS2 of section 3.2.1) is considered for the investigation of the latent space, considering raw data and variables normalized between 0 and 1. The input channels are C_p, M, ω, x and y.

Figure 3.4.1 shows the latent variables for $N_{ls} = 1$, $N_{ls} = 2$ and $N_{ls} = 3$ learned by the AE. The dataset consists of RANS computations of the flow around symmetric airfoil of different thickness at various angles of attack; effectively two free input parameters (thickness t and α) span the entire database.

By mapping the latent space to the free input parameters is clear that the AE is correctly or-

dering the solutions although t and α are not explicitly given as inputs and the training process is randomized. Figure 3.4.1a shows that $N_{ls} = 1$ is not sufficient to describe the entire dataset;



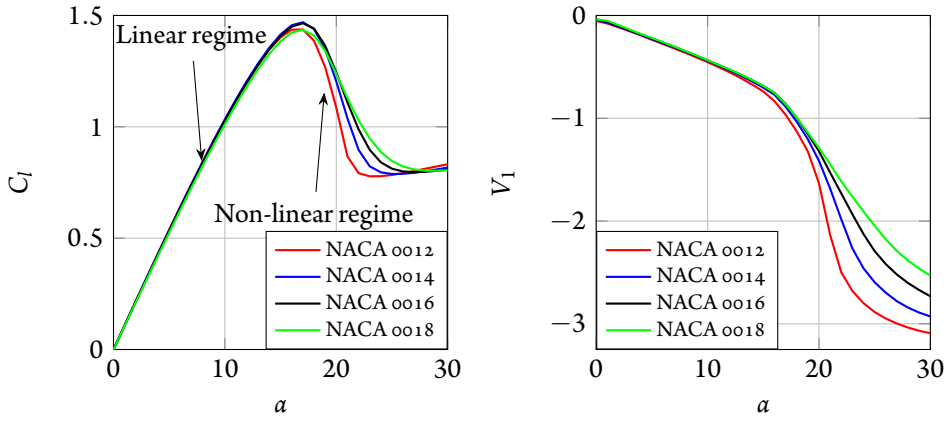
(a) $N_{ls} = 1$ latent space

(b) $N_{ls} = 2$ latent space

(c) $N_{ls} = 3$ latent space

Figure 3.4.1: $N_{ls} = 1, 2, 3$ latent spaces learned by the convolutional AE trained with the non-linear dataset.

for $\alpha < 10°$, one value of the latent variable corresponds to multiple airfoils. However, this case illustrates an important finding that connects the latent variables to the physics of the phenomenon. The AE infers that symmetrical airfoils have the same behaviour at low angle of attacks, while in the non-linear regime the thickness plays an important, differentiating role. This is confirmed by the analysis of the lift curves extracted from the RANS solutions (Figure 3.4.2a), which completely overlap for $\alpha < 10°$. The existance of a strong correlation between the lift coefficient C_l and the latent variable V_1 for $N_{ls} = 1$ is clearly visible also in Figure 3.4.3, where the C_l is reported as function of V_1. The different curves for the 4 airfoils collapse almost perfectly. In particular, the correlation between C_l and V_1 is greater than 90% in the linear regime.



(a) Lift curves, $Re_\infty = 5.0 \times 10^6$, $M_\infty = 0.15$, RANS simulations

(b) Latent space for $N_{ls} = 1$

Figure 3.4.2: Comparison between the lift curves of the airfoils in DS2 and the latent variables learned by the AE with the same dataset for $N_{ls} = 1$. $Re_\infty = 5.0 \times 10^6$, $M_\infty = 0.15$, SU2 RANS simulations using the SA turbulence model.

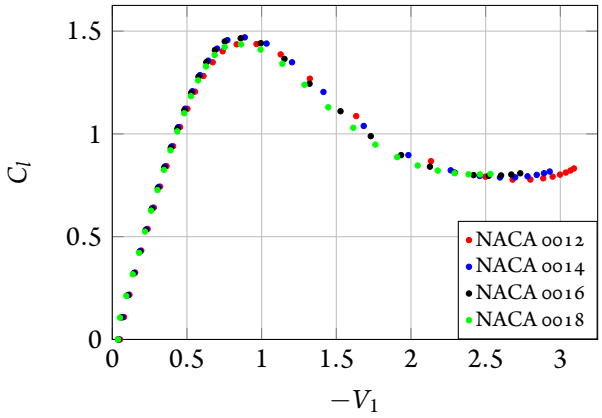


Figure 3.4.3: Lift coefficient C_l as function of the latent variable learned by the AE for $N_{ls} = 1$.

These are the first insights of the *physical imprinting* of the latent space.

3.5 Aerodynamic flow field predictions

The present investigation is informed by the previous published studies discussed in section 1.2.3, but it's aim is quite different: instead of constructing the AE to generate the flow given a geometrical representation of the airfoil, the AE is trained to reproduce the flow field itself

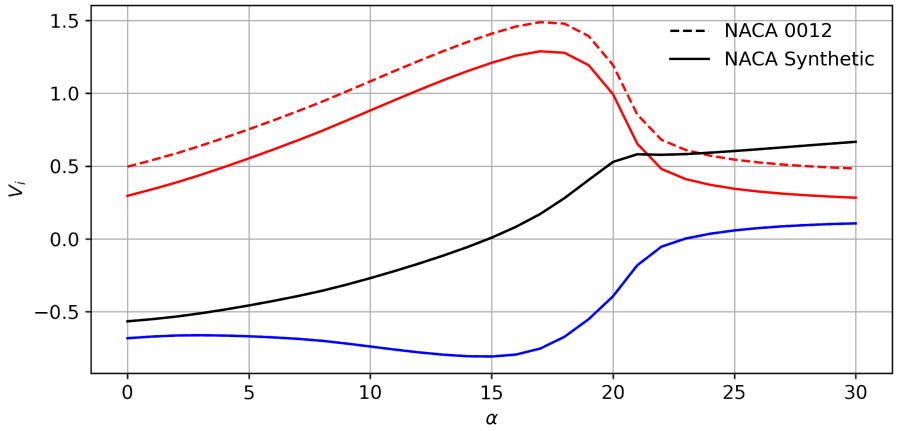


Figure 3.5.1: Translation of latent variables as function of the angle of attack α. Comparison with the latent variables of the NACA 0012 airfoil. V_1: —; V_2: —; V_3: — .

(input and output are the same) and then focused on the latent space representation to generate new airfoils and flow fields by using only the decoder. To have a clear idea on how the online deployment of the model works, a detailed description is provided in Appendix C.

3.5.1 Symmetric airfoils

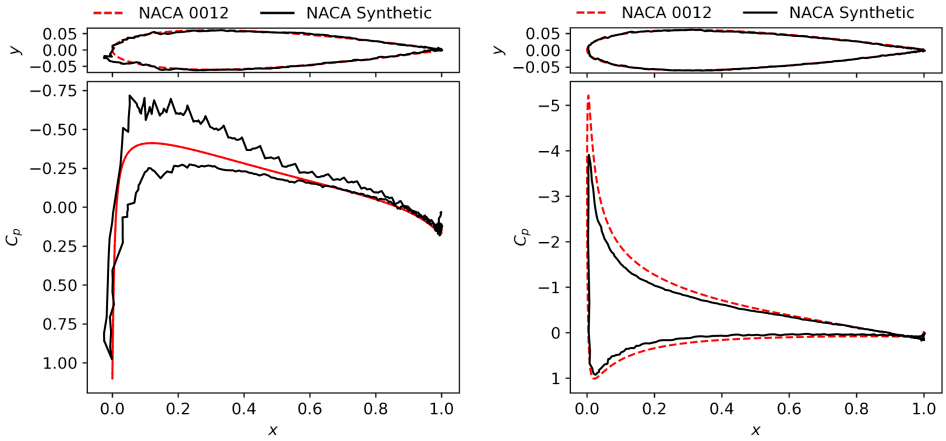
The selected database is the DS2 presented in section 3.2.1. The lift curves describing the physical phenomenon of DS2 are reported in Figure 3.4.2a. The phenomenon is linear for small angle of attack, while it becomes strongly non-linear when stall occurs.

The decoder element of the present algorithm can be used to quickly generate new *synthetic* solutions by modifying the values of the latent variables in input.

Latent variables translation

A simple modification to the latent variables is first attempted. The latent variables of the NACA 0012 were considered as the baseline. Figure 3.5.1 shows the variables as function of the angle of attack for the NACA 0012. A translation was applied to one of the components, V_3 by $\sim 10\%$. In Figure 3.5.2, the output of the decoder with the new latent variables is compared with the NACA 0012 baseline in terms of pressure distributions and airfoil geometry for 4 selected angles of attack ($\alpha = 0°,\ 10°,\ 20°$ and $30°$). As a first observation, it is clear that the synthetic geometry changes with the angle of attack, which can be interpreted as an indication that changes in the latent space variables cannot be arbitrary.

The decoder extracts an airfoil and a C_p distribution for $\alpha = 0°$ that corresponds to a negative angle of attack for a symmetric airfoil. The C_p curves and the airfoil geometries reported in



(a) $\alpha = 0°$ **(b)** $\alpha = 10°$

(c) $\alpha = 20°$ **(d)** $\alpha = 30°$

Figure 3.5.2: Comparison between the decoder output by giving in input the translated latent variables and the NACA 0012 baseline for $\alpha = 0°,\ 10°,\ 20°$ and $30°$.

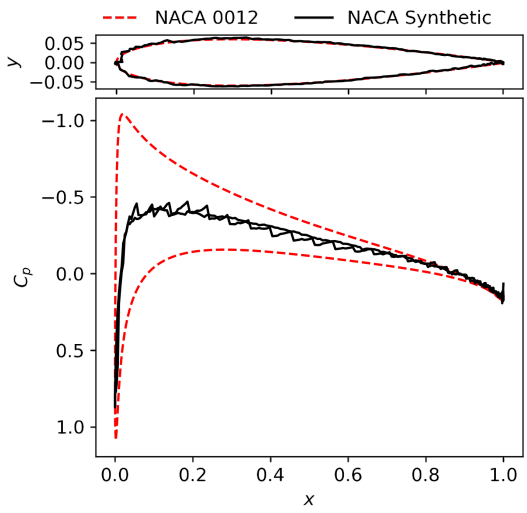


Figure 3.5.3: Output of the decoder with the translated variables at $a = 3°$.

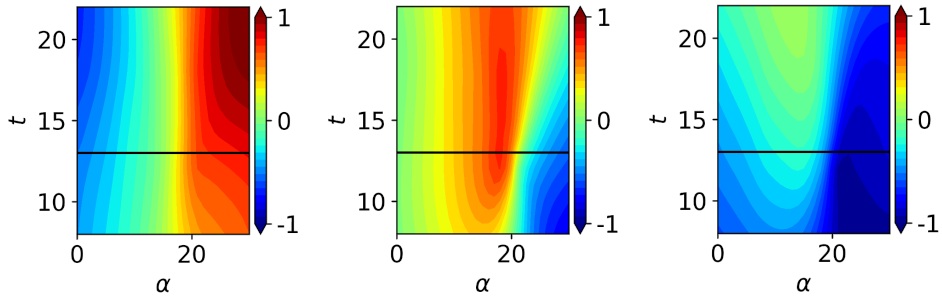
Figure 3.5.2 are obtained by extracting the first row of the reconstructed channels (C_p, x and y). The AE is trained giving in input $512 \times 170 \times 5$ tensors, while the airfoil geometry is the reconstruction of 2 vectors (x and y coordinates) of 512 elements each. This means that the airfoil geometry is only the 0.23% of the information that the AE is reproducing. In addition, the poor quality of the leading and trailing edge reconstructions are due to the presence of a thickening of the mesh grid in these regions, so the values of the coordinates are very close to each other and a small error influences the reconstruction quality.

Looking at the C_p of Figure 3.5.2, it seems that the translation of V_3 corresponds to a translation of a. In particular, considering the slope of V_3 in its linear part ($\frac{\Delta V_3}{\Delta a} \approx 0.065/\circ$), the translation here applied ($\Delta V_3 = -0.2$) corresponds to a new zero-lift angle of attack $a_{zl} = 3°$. Figure 3.5.3 shows that this is consistent with the results provided by the AE for $a = 3°$: the pressure coefficient on the upper and lower surfaces of the synthetic airfoil overlap, returning $C_l = 0$. The behaviour of V_3 resembles the lift curve, and the results corroborate this observation.

Interpolation in the latent space

The results just presented show that it is not possible to manipulate the latent space arbitrarily, and it is necessary to preserve the correlation between the latent variables. A controlled strategy is here developed to extract new synthetic airfoils and flow fields from the AE.

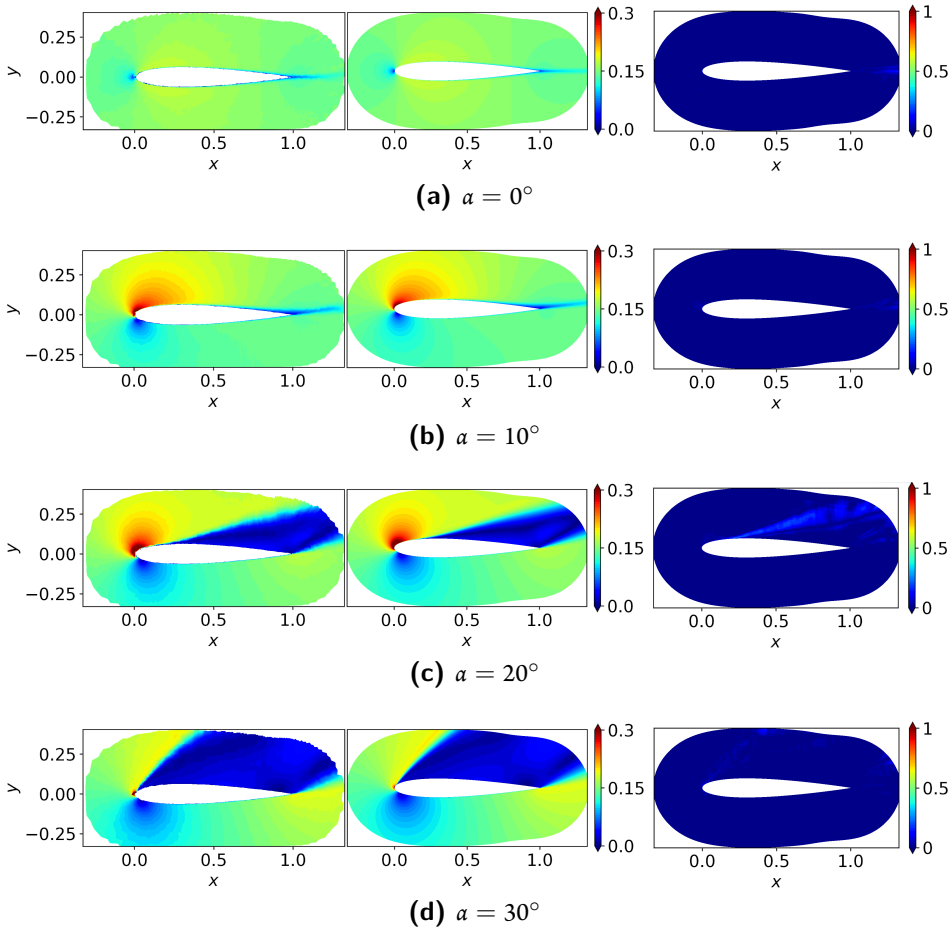
In the first step a mapping of the latent space on the database free parameters (airfoil thickness and angle of attack) is computed. The contour mapping is reported in Figure 3.5.4 for each latent variable of the latent space with $N_{ls} = 3$. This mapping facilitates easy interpolation, and the latent variables corresponding to 13% airfoil thickness are extracted (represented by the



(a) V_1 **(b)** V_2 **(c)** V_3

Figure 3.5.4: Contour mapping of the latent variables on the database parameters.

black line in Figure 3.5.4). Starting from these interpolated latent variables the decoder generates solutions that can also be compared to the RANS solutions generated for the same *expected* airfoil (NACA 0013). Figure 3.5.5 shows the Mach number contours for 4 angles of attack of the NACA 0013: reference solution (ground truth obtained from RANS simulation), decoder prediction and percentage error (computed with respect to the free-stream Mach number). Clearly, the AE accurately predicted the flow field around the new unseen airfoil, with a negligible percentage error. In order to have a more quantitative analysis of the result, it is interesting to look at the pressure coefficient C_p on the airfoil surface as reported in Figure 3.5.6. The dashed red curves refer to the NACA 0013 RANS solution, while the black curves refer to the decoder output (the synthetic NACA 0013), and they perfectly match both in the linear regime (with the attached flow in the first two plots of Figure 3.5.6) and the non-linear regime and stall conditions (the last two plots of Figure 3.5.6). Also the geometry computed by the decoder is closely matching the expected NACA 0013 airfoil, and it is not changed with the angle of attack, showing that a coherent latent space manipulation leads to a consistent result.



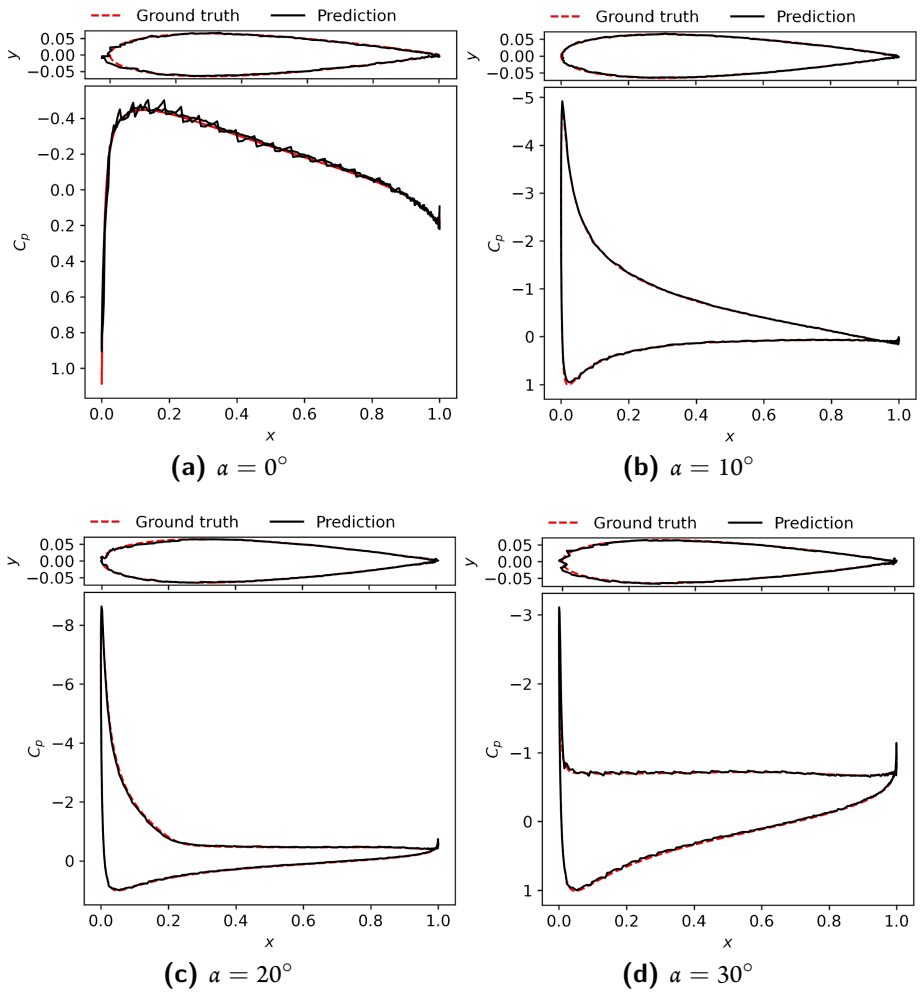
(a) $\alpha = 0°$

(b) $\alpha = 10°$

(c) $\alpha = 20°$

(d) $\alpha = 30°$

Figure 3.5.5: Mach number contours. From the left to the right: decoder prediction; ground truth; percentage error. For a NACA 0013 at $\alpha = 0°, 10°, 20°$ and $30°$.



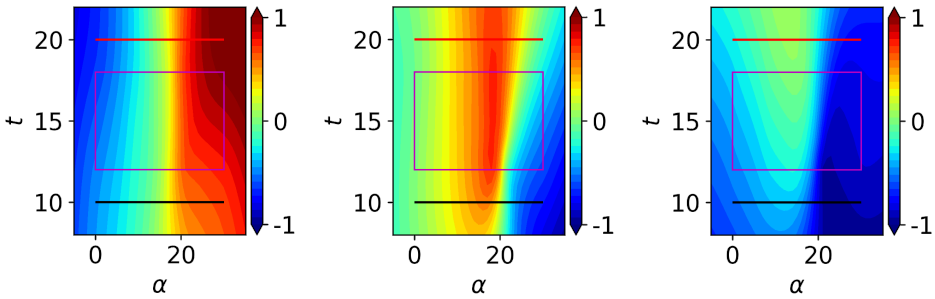
(a) $\alpha = 0°$ **(b)** $\alpha = 10°$

(c) $\alpha = 20°$ **(d)** $\alpha = 30°$

Figure 3.5.6: Comparison of the C_p on the airfoil surfaces for a NACA 0013 at $\alpha = 0°,\ 10°,\ 20°$ and $30°$.

Extrapolation beyond the latent space

The result presented in the previous section clearly shows that it is possible to interpolate in the latent space learned by an AE in order to obtain accurate aerodynamic predictions for a strongly non-linear phenomenon as the airfoil stall. But is it possible to extrapolate beyond the training dataset? A cubic extrapolation of the latent variables in the α-t plane was performed, obtaining the new contour maps in Figure 3.5.7, where the magenta rectangle represents the original boundaries of the input free parameters (the boundaries of the design-space of the training database). The values of the three latent variables at constant thickness $t = 20\%$ and $t = 10\%$



(a) V_1 **(b)** V_2 **(c)** V_3

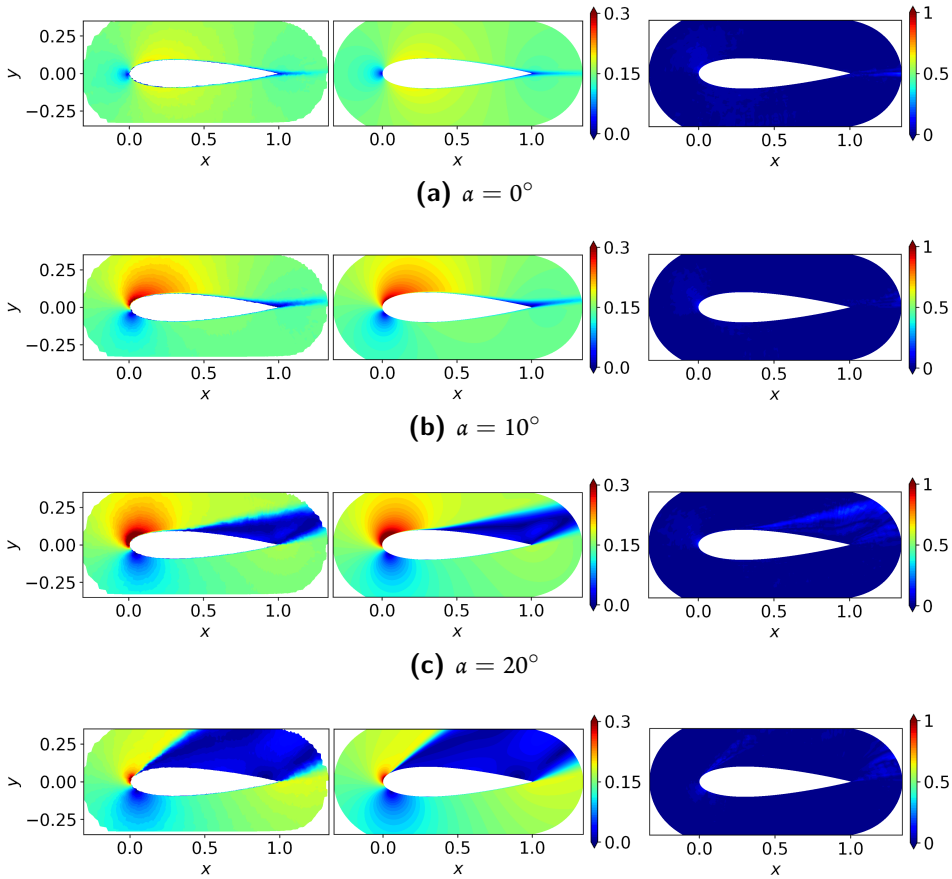
Figure 3.5.7: Extrapolated contour mapping of the latent variables on the database parameters. Magenta rectangle: boundaries of the interpolation region; black line: NACA 0010 extrapolation; red line: NACA 0020 extrapolation.

are extrapolated, respectively represented by the continuous red and black lines in Figure 3.5.7.

Figure 3.5.8 shows the Mach number contours (for $\alpha = 0°,\ 10°,\ 20°$ and $30°$) extracted from the RANS solutions of the NACA 0020 and the ones generated by the decoder with the extrapolated latent variables in input. The contour of the Mach number of the synthetic flow field appear to be in reasonable agreement with the RANS solution, however, in order to obtain a better evaluation of the results, the C_p on the airfoil surface is proposed in Figure 3.5.9. Once again, the output of the decoder is in agreement with the RANS solution for the NACA 0020 for both the linear and non-linear regimes.

Figure 3.5.10 and 3.5.11 show the result of the extrapolation for $t = 10\%$ compared with the CFD results. Also in this case the main physical phenomenon is captured by the AE.

Finally, an extrapolation in α at constant thickness $t = 12\%$ was performed. Figure 3.5.12 shows the extrapolation result for $\alpha > 30°$. The results highlight that as one moves farther from the training dataset, the accuracy gradually degrades (consider for instance the airfoil leading edge of the geometry reconstruction). In these extrapolatory cases the accuracy is somewhat degraded with the respect to the interpolation case; specifically the maximum expansion region (the negative c_p peak) is not exactly captured by the decoder in the strongly non-linear part of the phenomenon.



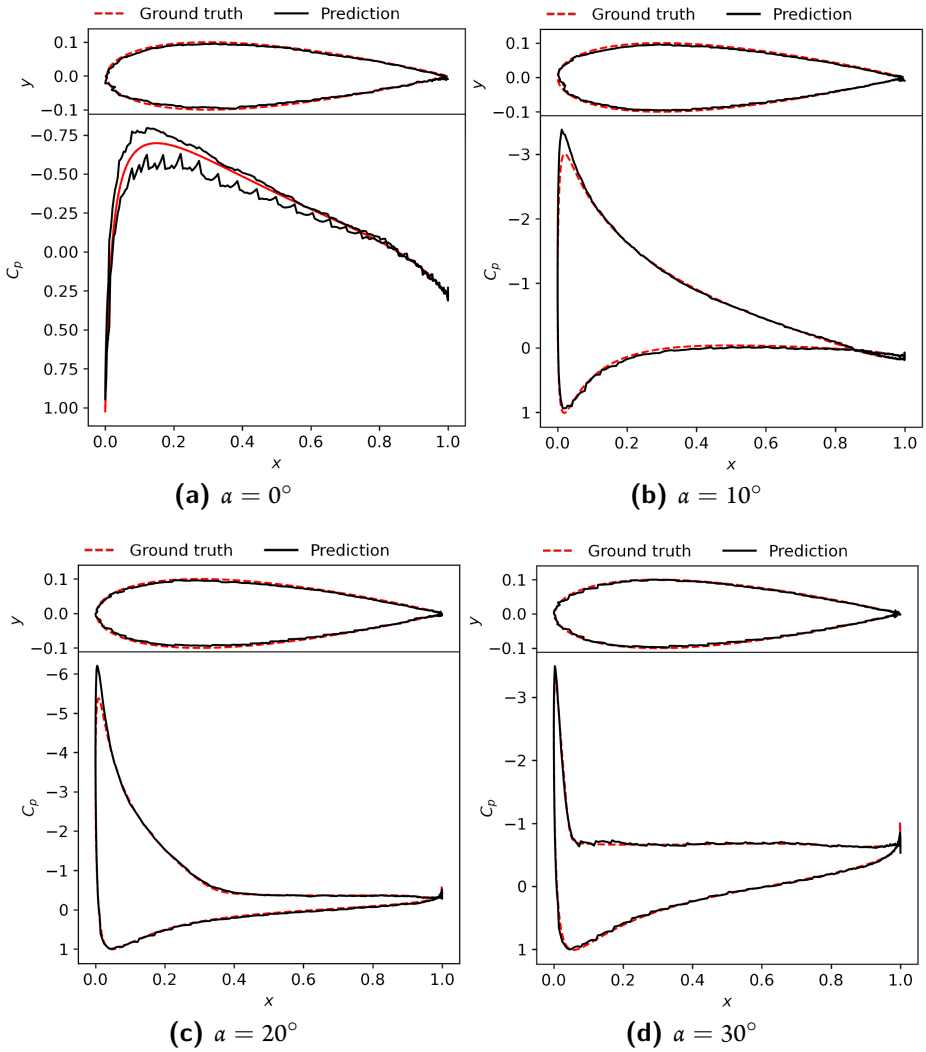
(a) $\alpha = 0°$

(b) $\alpha = 10°$

(c) $\alpha = 20°$

(d) $\alpha = 30°$

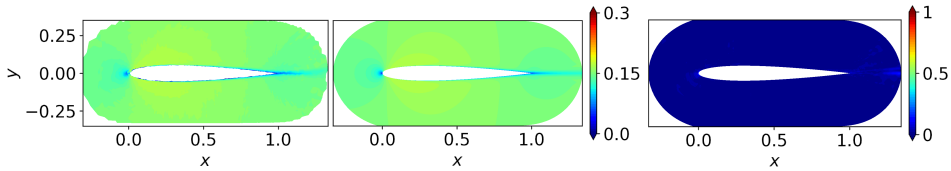
Figure 3.5.8: Mach number contours. From the left to the right: decoder prediction; ground truth; percentage error. For a NACA 0020 at $\alpha = 0°$, $10°$, $20°$ and $30°$. Airfoil extrapolated beyond the latent space.



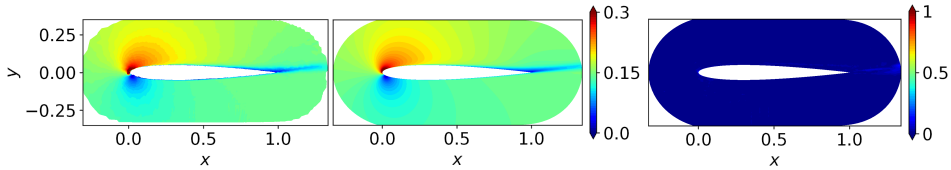
(a) $\alpha = 0°$ **(b)** $\alpha = 10°$

(c) $\alpha = 20°$ **(d)** $\alpha = 30°$

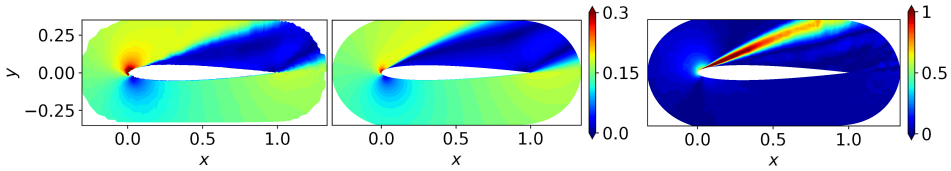
Figure 3.5.9: Comparison of the C_p on the airfoil surfaces for a NACA 0020 at $\alpha = 0°,\ 10°,\ 20°$ and $30°$. Airfoil extrapolated beyond the latent space.



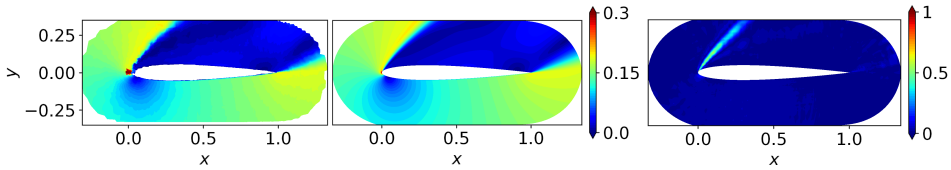
(a) $\alpha = 0°$



(b) $\alpha = 10°$

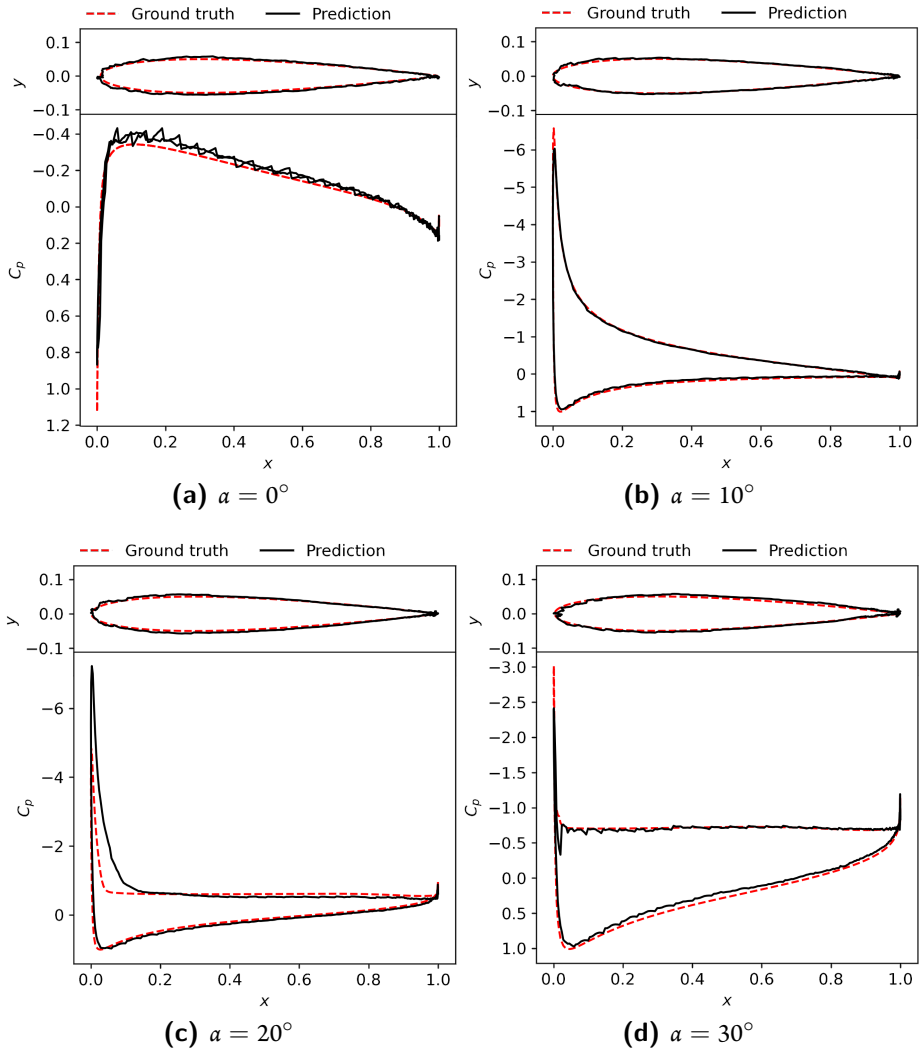


(c) $\alpha = 20°$



(d) $\alpha = 30°$

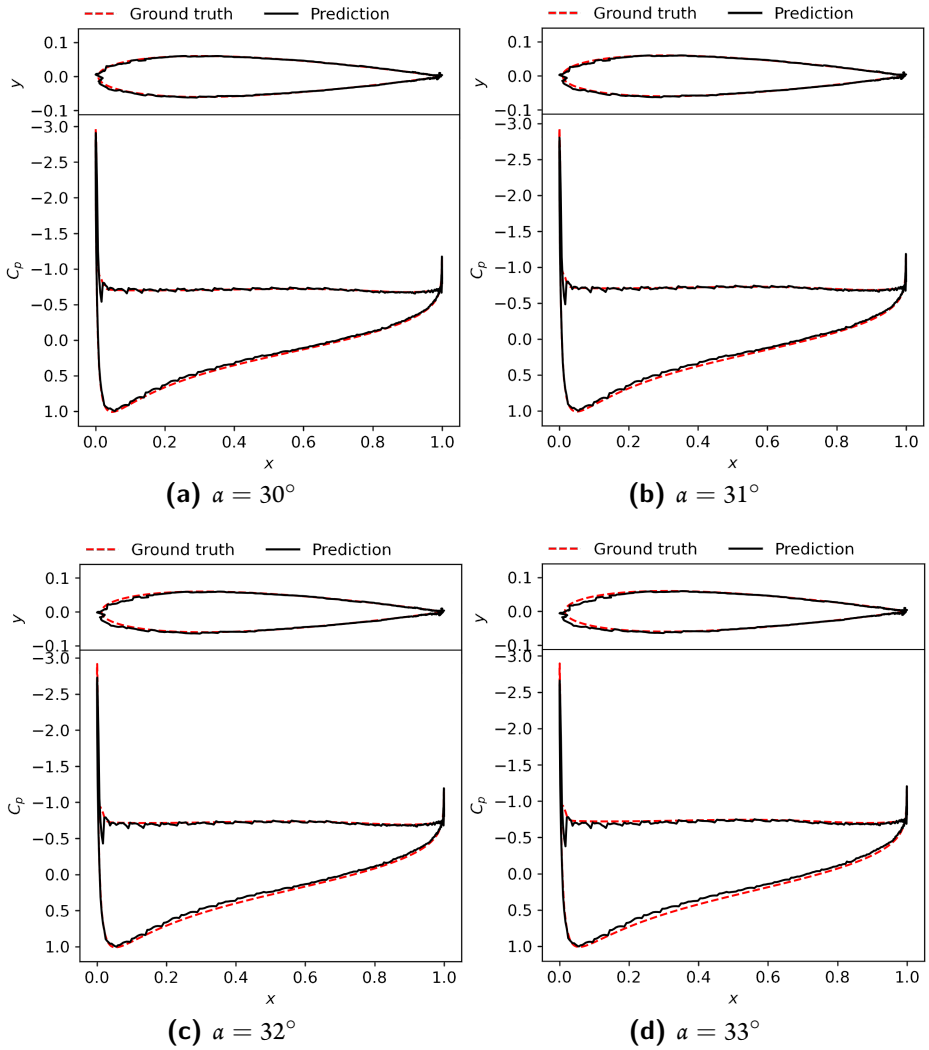
Figure 3.5.10: Mach number contours. From the left to the right: decoder prediction; ground truth; percentage error. For a NACA 0010 at $\alpha = 0°,\ 10°,\ 20°$ and $30°$. Airfoil extrapolated beyond the latent space.



(a) $\alpha = 0°$ **(b)** $\alpha = 10°$

(c) $\alpha = 20°$ **(d)** $\alpha = 30°$

Figure 3.5.11: Comparison of the C_p on the airfoil surfaces for a NACA 0010 at $\alpha = 0°,\ 10°,\ 20°$ and $30°$. Airfoil extrapolated beyond the latent space.



(a) $\alpha = 30°$ **(b)** $\alpha = 31°$

(c) $\alpha = 32°$ **(d)** $\alpha = 33°$

Figure 3.5.12: Comparison of the C_p on the airfoil surfaces for a NACA 0012 at $\alpha = 30°,\ 31°,\ 32°$ and $33°$. αs extrapolated beyond the latent space.

3.5.2 Non-symmetric airfoils

Theory of wing sections by Abbott and Doenhoff [84] is a critical resource to study airfoil aerodynamics and the dependency on both the geometry and the operating conditions (angle of attack and Reynolds number). The presented results are based on a vast collection of experimental data. In addition surface pressure distributions can be obtained for all NACA airfoils by a semi-empirical method based on tables collected in the book. This inspired the development of a ML approach that could leverage (a small number of) numerical simulations to produce a complete set of flow fields around NACA airfoils as function of angle of attach and Reynolds numbers. This extended Abbott approach demonstrates the potential of using ML as a powerful generator of solutions of engineering interest.

3 database parameters

The geometry of a NACA 4 digits airfoil can be identified by 3 numbers (4 digits D_i): $D_1D_2D_3D_4$, which denote the maximum percentage chamber $m = D_1/100$, position of maximum chamber $p_m = D_2/10$ and maximum percentage thickness $t = D_3D_4/100$.

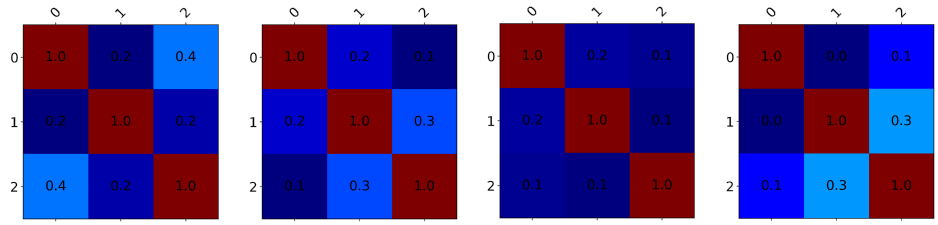
As first step, a database with 3 parameters is constructed: $\alpha \in [-30°, +30°]$; $m \in [0, 6]$; $t \in [5, 15]$. 90 cases are randomly selected as combination of the input parameters. In particular, the selected dataset is reported in table 3.5.1. The input to the AE are the primitive variables

NACA	$\alpha[°]$												
	-30	-25	-20	-15	-10	-5	0	5	10	15	20	25	30
0005		●				●	●		●	●			●
0008	●				●	●	●		●				●
0010		●	●	●			●			●		●	
0012		●						●	●	●	●		●
0016	●	●	●	●		●			●				
2405	●	●	●				●				●	●	
2410				●			●	●	●			●	●
2414	●	●			●		●			●	●		
2416		●			●	●		●	●				●
4408		●		●			●			●	●	●	
4412		●	●	●			●			●		●	
4416	●				●					●	●	●	●
6405			●			●	●		●	●	●		
6412	●	●	●	●				●					●
6416		●		●		●	●	●				●	

Table 3.5.1: Samples of the aerodynamic database. 3 database parameters: m, t and α.

and turbulence quantities: pressure p, velocity components u and v, density ρ and the Spalart variable $\widetilde{\nu}$. In this way the flow fliel is completely characterized.

The AE was trained using the loss function in eq.(3.2) which is a composition of the Mean Square Error between input and its reconstruction and a regularization term (L_2 norm) on the latent space. The regularization weight λ is a user-defined parameter. An analysis on its value was conduced, by training various AEs with different values of λ. Then, the latent space is investigated for each weight. In particular, Figure 3.5.13 shows the correlation matrices between the latent variables as function of λ. In the correlation matrices, the blue color represent a low correlation, while the red is representative of high correlation. To obtain an index of the correlation



(a) $\lambda = 0$ **(b)** $\lambda = 2$ **(c)** $\lambda = 5$ **(d)** $\lambda = 10$

Figure 3.5.13: Correlation matrices of the latent variables as function of λ. Blue means low correlation, red high correlation.

level, it is possible to compute the determinant of the correlation matrix. Figure 3.5.14 shows a comparison between the correlation index and the MSE for different λs. It is a measure of how

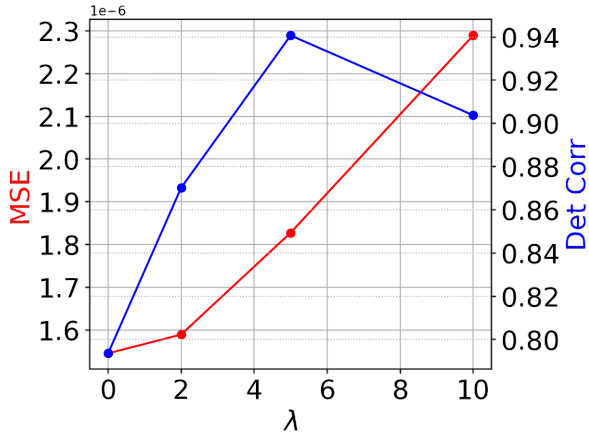


Figure 3.5.14: Correlation index and MSE as function of λ.

much the variables are linearly related to each other. A determinant of 0 indicates that the variables are perfectly linearly dependent (one variable can be expressed as a linear combination of the others), while a determinant of 1 indicates that the variables are uncorrelated (orthogonal).

Disentangled latent variables are more feasible to be interpreted, however, while the determinant grows with λ, the MSE also increase, leading to a degradation of performance. $\lambda = 5$ was selected in what follows.

The disentanglement of latent variables within the latent space of an AE has been explored in the literature using the previously mentioned β-VAE [54]. In this framework, latent variables are treated as random variables defined by both a mean and a variance. In this case, the latent vector comprises random variables, typically assumed to follow a Gaussian distribution with independent and identically distributed properties. Once the learning phase is concluded, generating realizations from the stochastic latent space allows for the creation of a probabilistic representation of $\hat{\mathbf{x}}$. To briefly explain, the training of a β-VAE involves using a loss function that combines the reconstruction error and the *Kullback-Leibler Divergence* (KL divergence) [85]. This divergence is weighted with a hyperparameter denoted as β, which regulates the trade-off between reconstruction accuracy and the degree of disentanglement in the learned latent representations:

$$\text{KL divergence} = -0.5 \sum_{i=1}^{N_{ls}} \left(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2\right) \tag{3.3}$$

where μ_i and σ_i are the mean and the variance of the i-th latent variable. The regularization term introduced in the loss function encourages the approximate posterior distribution to closely resemble a chosen prior distribution, typically a multivariate Gaussian distribution with a mean of 0 and a variance of 1. Previous research, as documented in [52, 78], has showed that increasing the hyperparameter β leads to disentangled latent variables. In the present context of *deterministic* AEs, the latent variables are real numbers rather than Gaussian random variables. Therefore, the application of the KL divergence is not applicable. However, it is worth noting that introducing a regularization term on the L_2 norm of the latent space, weighted by a factor λ (second term of eq.(3.2)), has been shown to yield similar outcomes, as elaborated in this study.

The obtained latent space for the training dataset is reported in Figure 3.5.15. It is interesting

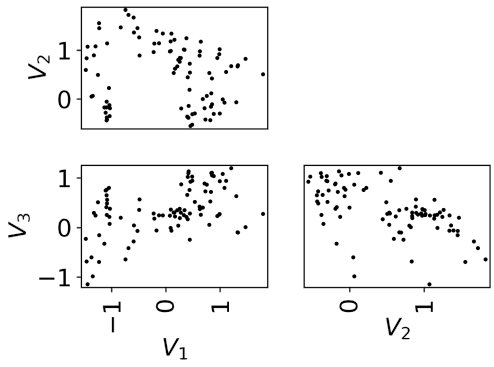


Figure 3.5.15: Latent space projections.

to color the latent variables as function of the aerodynamic force coefficients (C_l and C_d). The

result is reported in Figure 3.5.16. Clearly the latent space is sorted in terms of lift and drag.

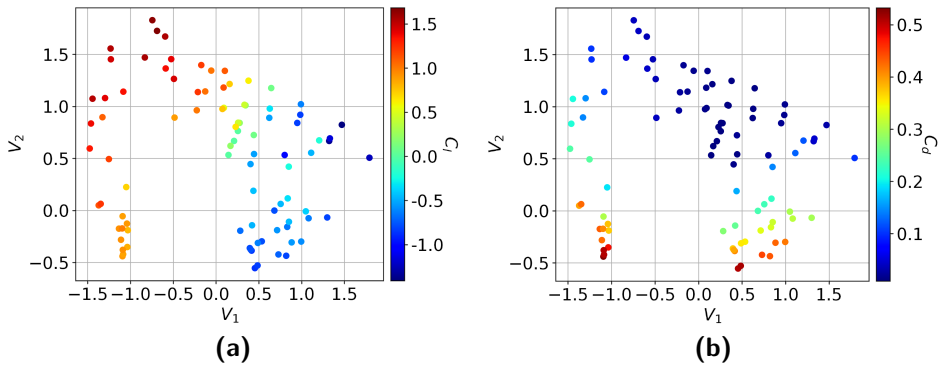


Figure 3.5.16: Latent space projections, colored with the aerodynamic force coefficient. (a): C_l; (b): C_d.

Once again, a *physical imprinting* is found in the latent space.

At this point it is possible to map the latent space on the database parameters as already described in section 3.5.1. The selected case to investigate is the NACA 4408. The extracted latent variables are reported in Figure 3.5.17. These latent variables, as function of α (one of the dataset

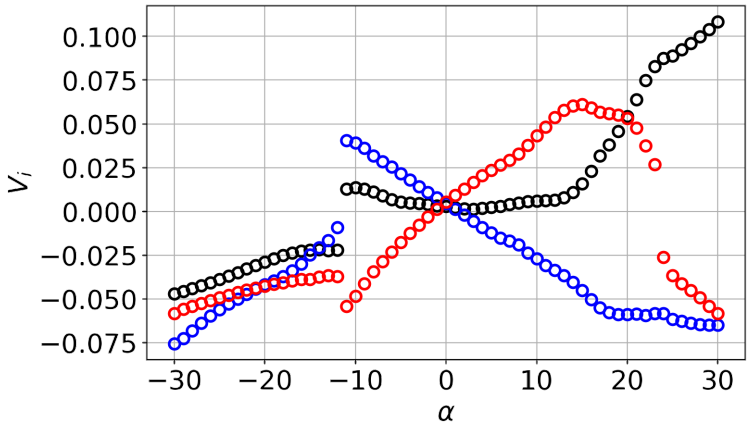
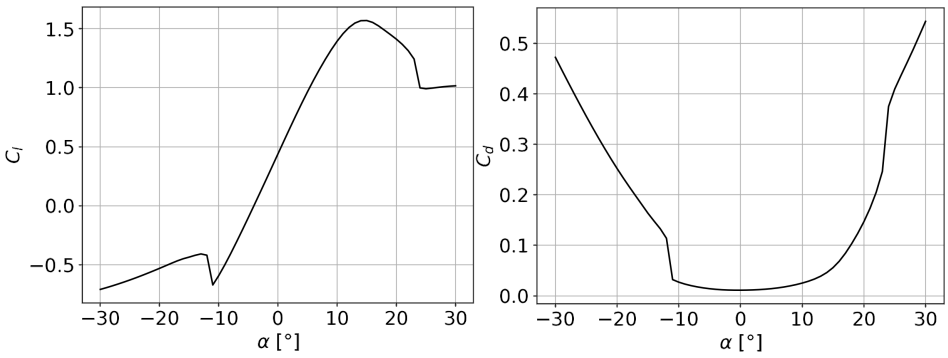


Figure 3.5.17: Latent variables as function of the angle of attack for a NACA 4408. ○: V_1;○: V_2; ○: V_3.

parameters), present some discontinuities at $\alpha = -11°$ and $\alpha = 23°$. The interesting observation, is that such jumps reflect the discontinuities present in the aerodynamic coefficients of the NACA 4408 airfoil (see Figure 3.5.18). The latent variables of Figure 3.5.17 can be fed into the decoder to generate new flow fields. In particular, the necessity here is to verify that the prediction corresponds to a RANS solution of a NACA 4408 airfoil. The decoder output is a tensor which contains all the physical variables used during the training (p, u, v, ρ and $\widetilde{\nu}$). The results



(a) Lift coefficient **(b)** Drag coefficient

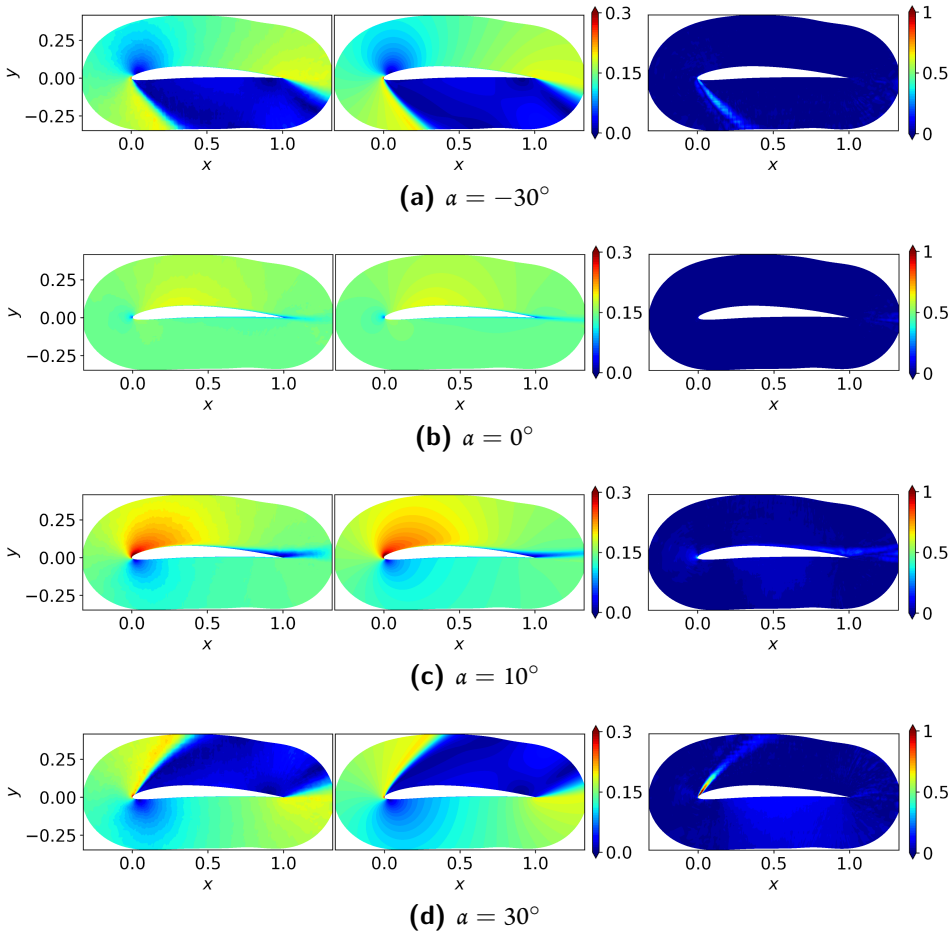
Figure 3.5.18: Aerodynamic coefficients of a NACA 4408 airfoil. RANS simulation using SU2. $Re_\infty = 5 \times 10^5$, $M_\infty = 0.15$, SA turbulence model.

obtained are here reported in terms of contour maps of the local Mach number (Figure 3.5.19) and surface pressure coefficient 3.5.20). The results are reported for $\alpha = -30°$ for the negative deep stall regime; $\alpha = 0°$ and $\alpha = 10°$ in the linear region of the phenomenon and $\alpha = 30°$ which is located in the strongly non-linear part of the phenomenon.

For a more general evaluation of the results, the lift curves are compared by computing the lift coefficient as follow:

$$\overline{C_l} = \frac{2}{\rho_\infty V_\infty^2} \int_0^1 (p_l - p_u) \, \mathrm{d}\left(\frac{x}{c}\right) \tag{3.4}$$

where p_u and p_l are the upper and lower airfoil surface pressure, respectively. This is an approximate computation of the lift coefficient ($\overline{C_l}$) since it takes into account only the pressure distribution on the airfoil surface. The comparison between the ground truth and the decoder prediction is reported in Figure 3.5.21. The results obtained by the AE are in close agreement with the RANS simulations. Some small oscillations are present in the AE predictions given by the fact that the training database is very coarse (considering the complexity of the problem). However, the AE is able to capture the strongly non-linear phenomenon and to produce reliable and accurate results in terms of physical variables in the whole flow field. This is possible with a coarse dataset and performing an extreme data compression to just 3 latent variables. In the next section the effect of the database dimension is accounted in a more extensive manner.



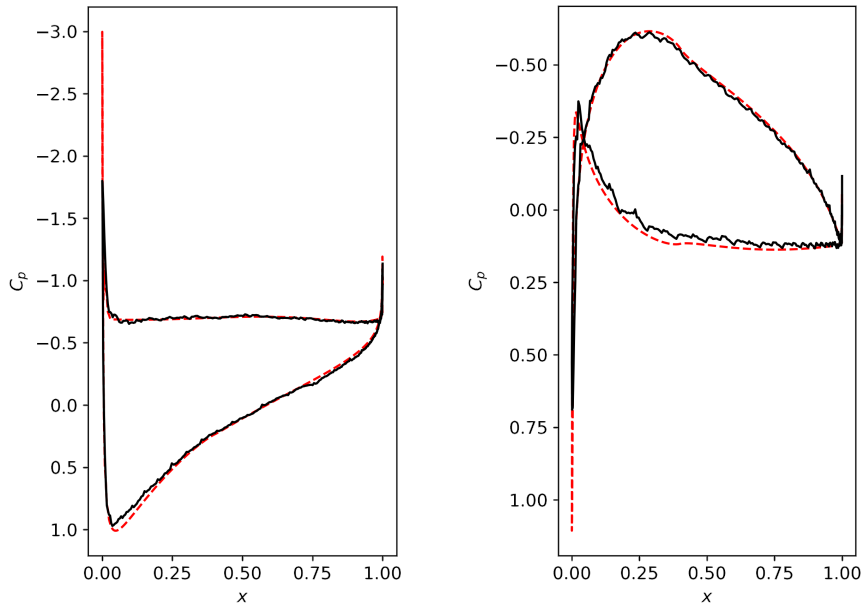
(a) $\alpha = -30^\circ$

(b) $\alpha = 0^\circ$

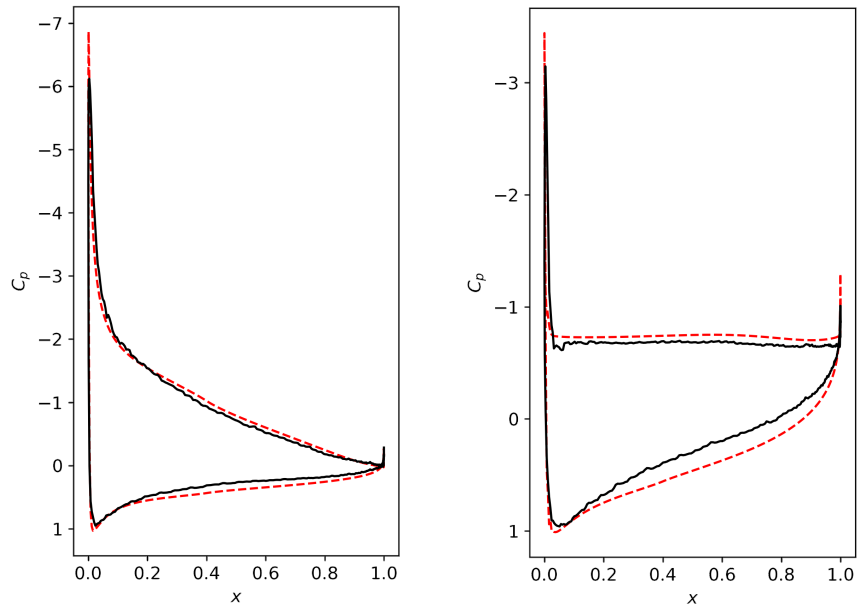
(c) $\alpha = 10^\circ$

(d) $\alpha = 30^\circ$

Figure 3.5.19: Mach number contours. From the left to the right: decoder prediction; ground truth; percentage error. For a NACA 4408 at $\alpha = -30^\circ$, 0°, 10° and 30°.



(a) $\alpha = -30°$ **(b)** $\alpha = 0°$



(c) $\alpha = 10°$ **(d)** $\alpha = 30°$

Figure 3.5.20: Comparison of the C_p on the airfoil surfaces for a NACA 4408 at $\alpha = -30°,\ 0°,\ 10°$ and $30°$ (dataset extrapolation region).

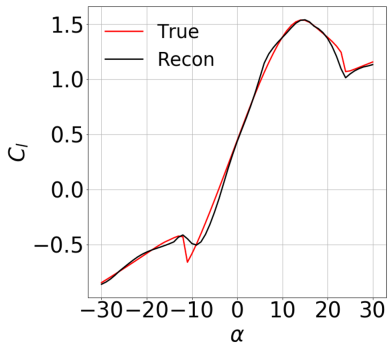


Figure 3.5.21: Lift curve of the NACA 4408 airfoil at $Re_\infty = 5 \times 10^5$, $M_\infty = 0.15$, SA turbulence model. Comparison between ground truth (*true*) and decoder prediction (*recon*).

5 DATABASE PARAMETERS

In this case, the complexity of the problem is increased by adding 2 more parameters in the database. An initial coarse database is constructed by selecting few cases by changing:

- $m \in [0;\ 6]$;
- $p_m \in [2;\ 5]$;
- $t \in [6;\ 22]$;
- $a \in [-30°;\ 30°]$;
- $Re_\infty \in \left[3 \times 10^6;\ 9 \times 10^6\right]$.

The selection of the initial database was performed using a traditional quasi-random strategy, namely Latin Hypercube Sampling (LHS) [86] that ensures sparsity and coverage of the space of the input parameters. Figure 3.5.22 shows the discrepancy index on the parameter space as function of the number of selected cases by the quasi-random algorithm. The *Centered Discrepancy* defined in [87] was selected. The lower the discrepancy is, the better the coverage of the parameter space is. The input physical variables are the primitive variables (unknown of the

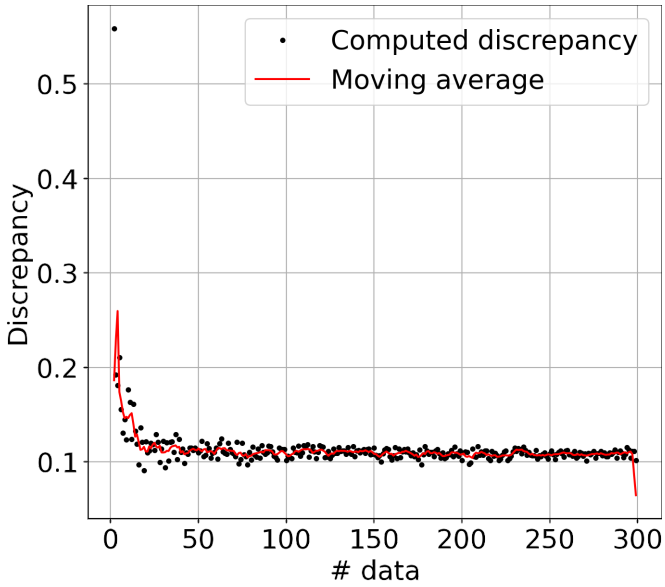


Figure 3.5.22: Discrepancy score on the parameter space as function of the number of selected cases by the quasi-random algorithm.

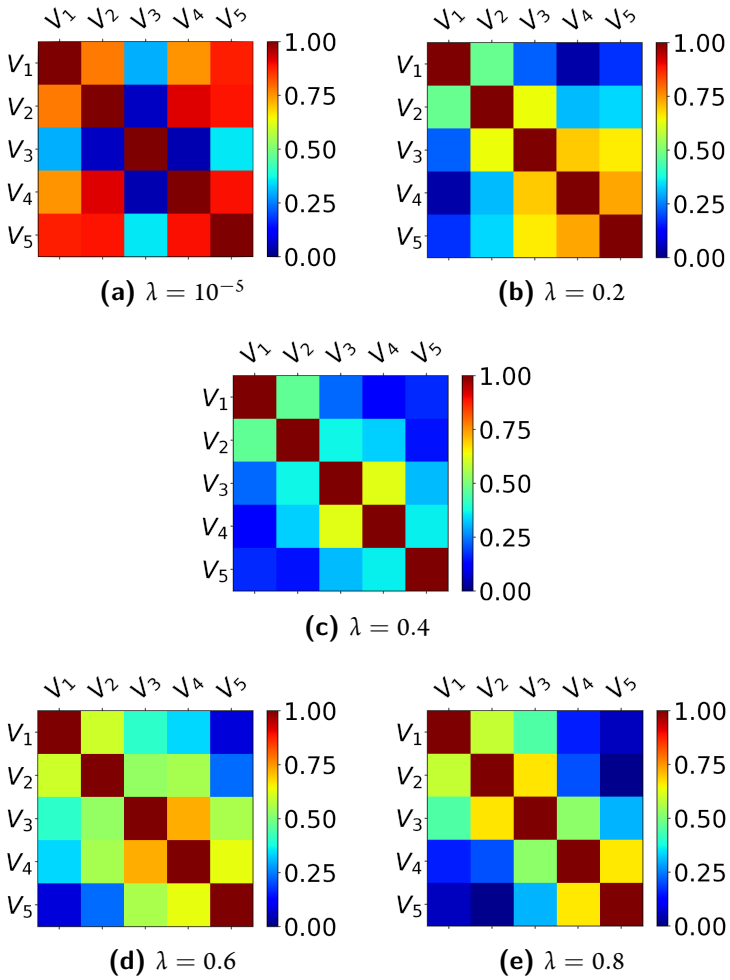
RANS equations): p, ρ, u, v and $\hat{\nu}$. Where p is the static pressure, ρ is the density, u and v are the velocity components in x and y directions and $\hat{\nu}$ is the Spalart variable. Each case is composed by a tensor which dimensions are $512 \times 256 \times 5$ (respectively elements in i direction, in j direction and number of channels, i.e. input variables).

An alternative approach based on known aerodynamic behavior of these airfoil would have been to limit the number of simulations at low angles of attack. However, the initial choice was to begin with quasi-random cases and subsequently refine the database in a second step, in order to show that it is possible to build a targeted database also in the absence of prior knowledge.

In order to describe the database, the selected number of latent variables N_{ls} needs to be at least equal to the number of the database parameters. In the present case, the database is composed by 5 parameters, therefore, a latent space dimension $N_{ls} \geq 5$ is needed. The aerodynamic database is randomly splitted for constructing the training and evaluation datasets. In particular, 15% of the solutions are used for evaluation and the input dataset is completely randomized with a batch size equal to 2. The AE is trained using a loss function composed by the mean squared error (MSE) between the input and the reconstruction, and an L_2 norm regularization on the latent space (eq. (3.2)). After a sensitivity analysis, $\lambda = 0.4$ is chosen. In particular, Figure 3.5.23 shows the correlation matrices between the latent variables by training the AE with different values of $\lambda = 10^{-5},\ 0.2,\ 0.4,\ 0.6,\ 0.8$. Where blue is low correlation and dark-red is high correlation. Figure 3.5.24 shows the loss function and the determinant of the correlation matrix as function of λ. The determinant of the correlation matrix is known to be related to the disentanglement of the latent variables. The more the determinant is close to 1, the more the latent variables are disentangled enabling a more clear interpretation. Comparing this result with the previous one of Figure 3.5.14 (with 3 database parameters), it is not possible to reach the same high levels of disentanglement of the latent variables. The more the complexity of the problem grows, the more it is difficult to orthogonalize the latent space. Figure 3.5.24, clearly shows that the best compromise between the reconstruction performance and the latent space disentanglement is obtained with $\lambda = 0.4$.

Figure 3.5.25 shows the projections of the latent space, learned by the AE, along the 5 latent variables, and colored by the free-stream Reynolds number Re_∞. It is clear that one parameter emerging from the latent space is the free-stream Reynolds number, in particular, the latent variable V_2 varies as the Reynolds number: V_2vsV_5 shows that 4 different clusters can be identified, each corresponding to the same value of Re_∞ (the database is composed by 4 different values of Re_∞). Of course, there cannot be a perfect correspondence between the latent variables and a physical parameter since there is not a complete disentanglement between the latent variables. To perform interpolation or extrapolations in the latent space of new cases, the latent variables were mapped on the database parameters normalized with respect the maximum absolute value of each parameter (as already described above). In this specific case, 5 parameters and 5 latent variables. Figure 3.5.26 shows the latent variables interpolated/extrapolated on the database parameters. In particular, since it is difficult to visualize the interpolation/extrapolation in 5 dimensions, in Figure 3.5.26, the mapped latent variables are presented as function of pairs of parameters and fixed the remaining ones. In particular, the fixed parameters are relative to NACA 2412 at $\alpha = 0^\circ$ and $Re_\infty = 5 \times 10^6$. Each square in the plots is a latent variable mapped with the RBF function on the database parameters, that is a slice of the 5-dimensional latent space.

At this point, with the same procedure previously described, it is possible to extract the latent variables of a specific case of interest, to generate a prediction of the flow field with the decoder. An example of application is here reported for two airfoils: **NACA 0012** at $Re_\infty = 5 \times 10^6$; and a **NACA 4420** at $Re_\infty = 7 \times 10^6$. Figure 3.5.27 shows a comparison between the original latent variables, with the interpolated latent variables as function of α for the NACA 4420. The dots are the latent variables obtained with the training, the continuous lines are the interpolated latent variables and the circles are the latent *correct* variables provided by the encoder when giving the entire solution in input (only for validation). It is interesting to notice that while the *correct* variables present the physical imprinting (the latent variables as function of the angle of attack follows the linearity/non-linearity of the physical phenomenon), the interpolated ones are very oscillatory. This difference is due to a lack of data for the NACA 4420 at $\alpha > 0°$. Regarding the AE predictions, it can be useful to compute the lift curve (as in eq. (3.4)), to see if the AE "learned" the physical phenomenon. Figure 3.5.28 shows the comparison between the lift curve computed on the ground truth and the one computed on the flow field predicted by the AE. The accuracy of the results is clearly unsatisfactory. The selected database is too coarse to handle a problem of this complexity: a strongly non-linear phenomenon with five design parameters varying within the dataset. For this reason, it is necessary to look for a strategy to refine the database.



(a) $\lambda = 10^{-5}$ **(b)** $\lambda = 0.2$

(c) $\lambda = 0.4$

(d) $\lambda = 0.6$ **(e)** $\lambda = 0.8$

Figure 3.5.23: Correlation matrices of the latent variables by training the AE with $N_{ls} = 5$ and changing λ.

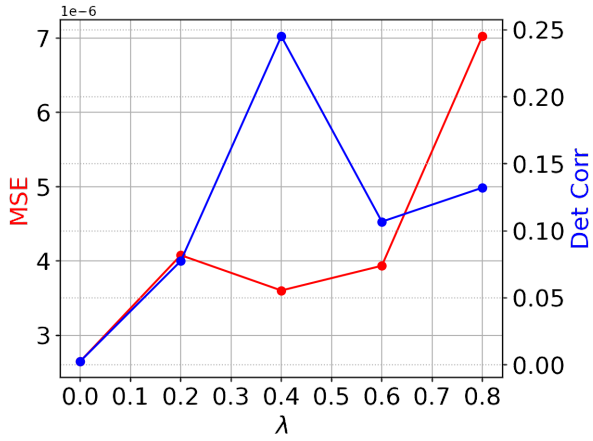


Figure 3.5.24: Correlation index and MSE as function of λ.

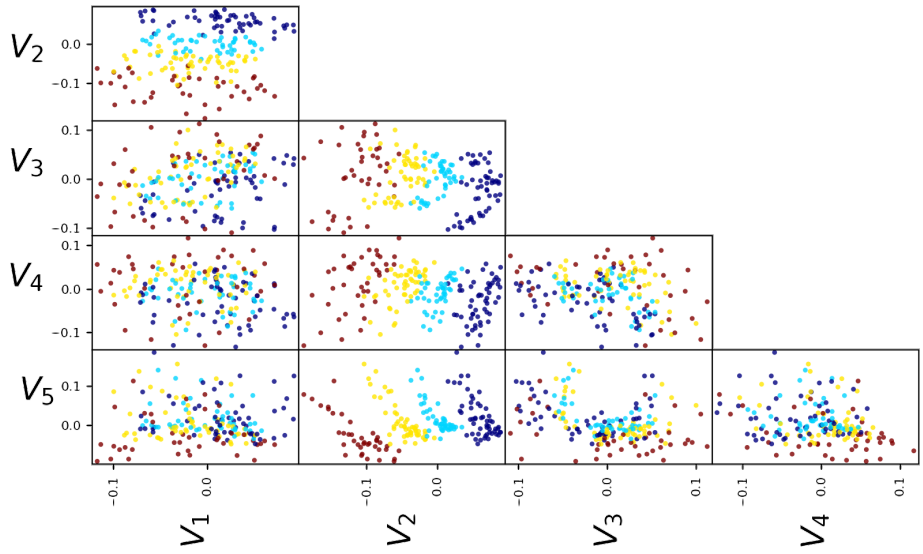
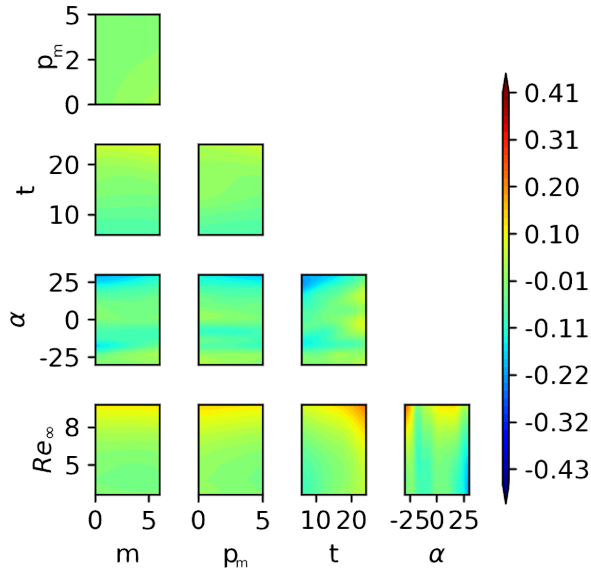
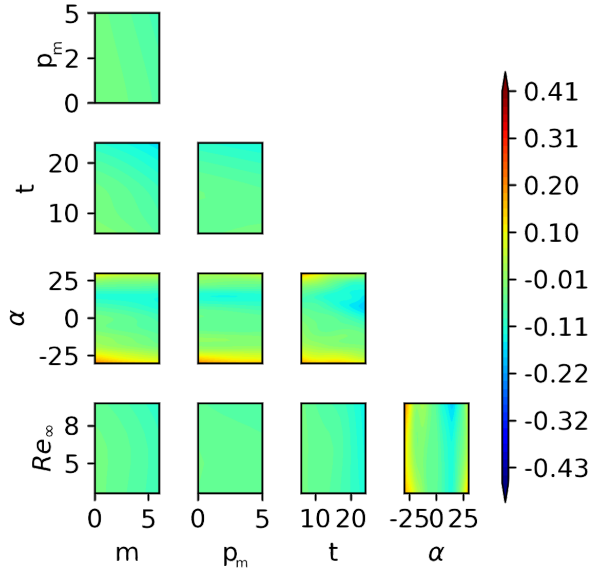


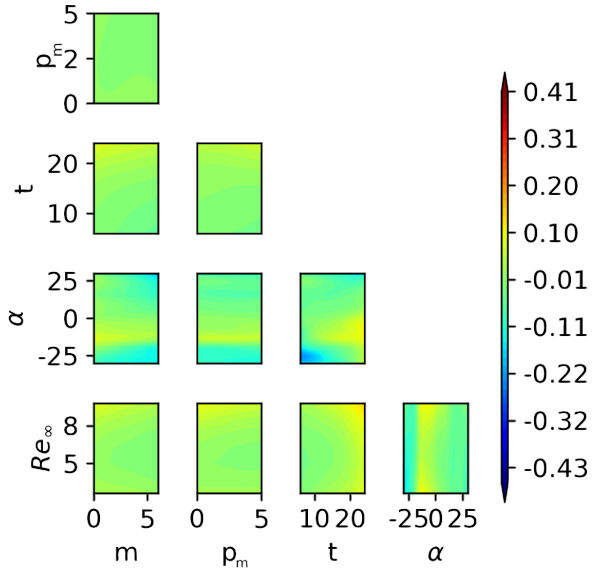
Figure 3.5.25: Latent space projections. Re_∞: ●: 3×10^6; ●: 5×10^6; ●: 7×10^6; ●: 9×10^6.


5
2
0
p_m
20
10
t
25
0
-25
α
8
5
Re_∞
0 5
0 5
10 20
-25 0 25
m
p_m
t
α
0.41
0.31
0.20
0.10
-0.01
-0.11
-0.22
-0.32
-0.43

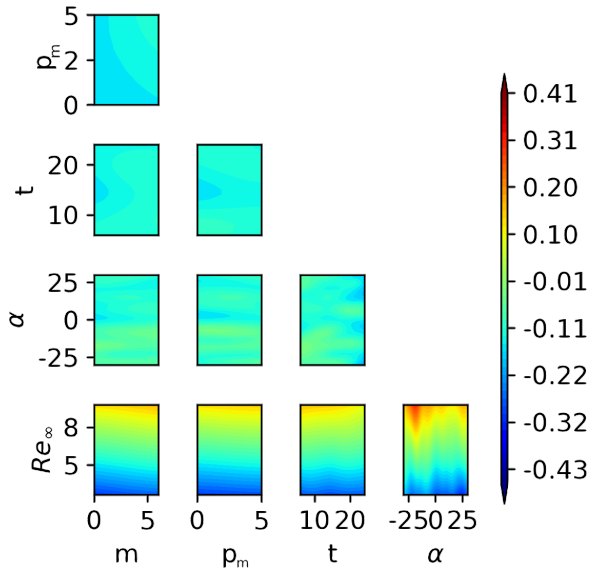
(a) V_1


5
2
0
p_m
20
10
t
25
0
-25
α
8
5
Re_∞
0 5
0 5
10 20
-25 0 25
m
p_m
t
α
0.41
0.31
0.20
0.10
-0.01
-0.11
-0.22
-0.32
-0.43

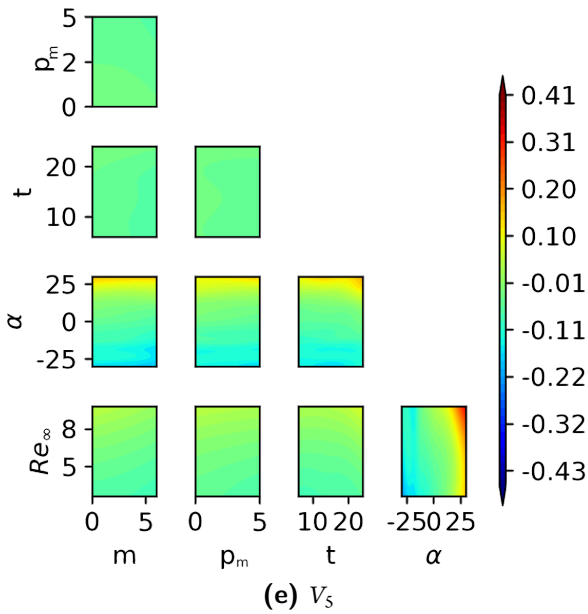
(b) V_2


5
2
0
p_m
20
10
t
25
0
-25
α
8
5
Re_∞
0 5
0 5
10 20
-25 0 25
m
p_m
t
α
0.41
0.31
0.20
0.10
-0.01
-0.11
-0.22
-0.32
-0.43

(c) V_3


5
2
0
p_m
20
10
t
25
0
-25
α
8
5
Re_∞
0 5
0 5
10 20
-25 0 25
m
p_m
t
α
0.41
0.31
0.20
0.10
-0.01
-0.11
-0.22
-0.32
-0.43

(d) V_4



(e) V_5

Figure 3.5.26: Latent space mapping on the database parameters scaled with respect to the maximum absolute value.

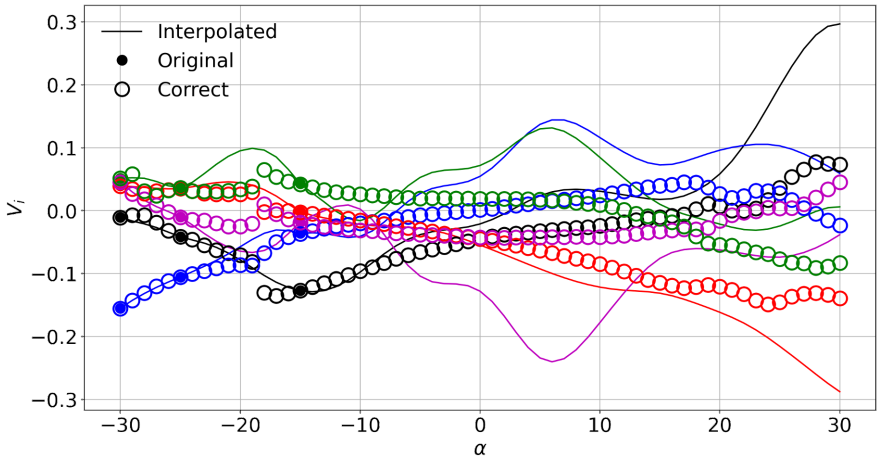
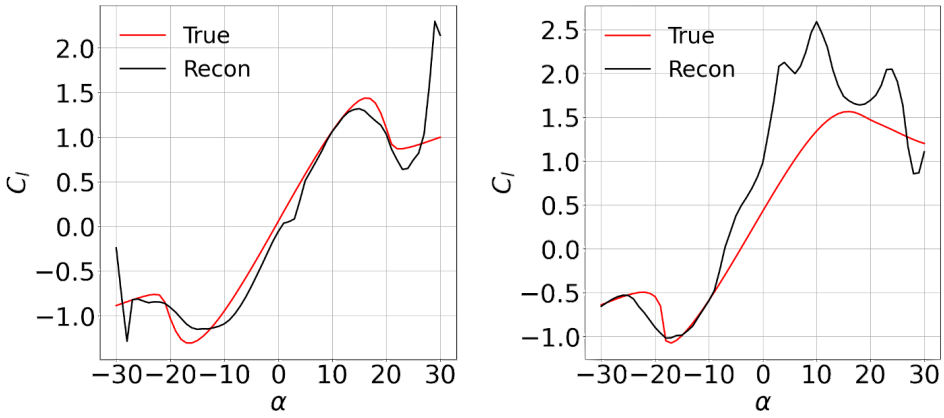


Figure 3.5.27: Latent variables as function of α for a NACA 4420. Comparison between the trained latent variables and the interpolated ones. –: V_1; –: V_2; –: V_3; –: V_4; –: V_5.



(a) NACA 0012 **(b)** NACA 4420

Figure 3.5.28: Lift curves of the NACA 0012 at $Re_\infty = 5 \times 10^6$ and the NACA 4420 at $Re_\infty = 7 \times 10^6$. Comparison between the ground truth (denoted with "true") and the AE prediction (denoted with "recon").

3.6 Adaptive database generation

A well known limitation of ML algorithms is their need for large datasets. For this reason, a strategy to limit the number of database cases is needed. This aspect is crucial in fields where data availability is constrained by the substantial computational efforts required to generate data, as is often the case in CFD. As already discussed, after training the AE, the latent variables as function of the database parameters are representative of the physical behaviour of the investigated phenomenon. For this reason, the parameters necessitating a database refinement were identified by selecting the maximum gradients of the interpolated latent variables. Figure 3.6.1 shows the

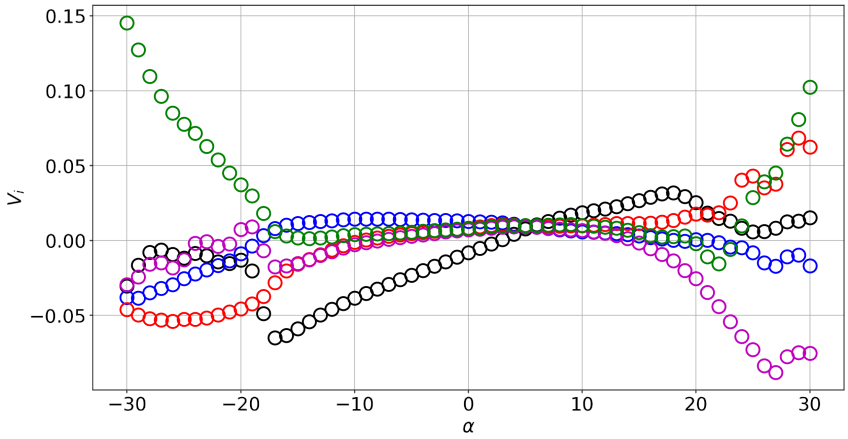
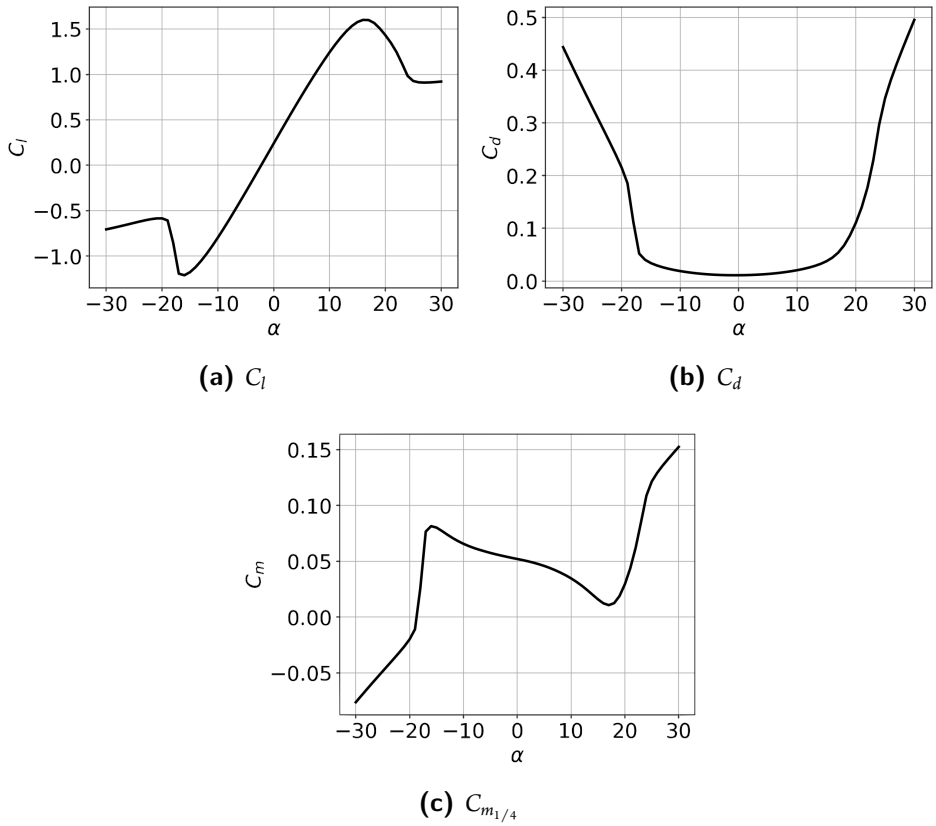


Figure 3.6.1: Latent variables for the NACA 2412 airfoil. ○: V_1; ○: V_2; ○: V_3; ○: V_4; ○: V_5.

latent variables as function of α using $N_{ls} = 5$, for a NACA 2412 at $Re_\infty = 5 \times 10^6$, provided by the encoder when giving in input the whole RANS simulations for all the angles of attack. The physical imprinting of the latent space is clearly visible by comparing the latent variables of Figure 3.6.1 with the aerodynamic coefficients in Figure 3.6.2: the jumps between $\alpha = -18°$ are present both in the latent variables and in the aerodynamic coefficients. Moreover for low angles of attack where the phenomenon is linear, also the latent variables have a linear behaviour, while they present strong non-linearities where the phenomenon becomes strongly non-linear (aerodynamic stall). It could be beneficial for the AE's training to include more information in regions where the description of the phenomenon is particularly challenging. In this case, a refinement around $\alpha = -18°$. To detect strong variations in the latent space, the gradient of the latent variable with respect to the angle of attack is considered. However, considering only the gradients in the latent space, does not take into account the presence of already existing data, therefore it would also select new cases very close to the already existing ones, potentially causing a biasing



(a) C_l **(b)** C_d

(c) $C_{m_{1/4}}$

Figure 3.6.2: Aerodynamic coefficients for a NACA 2412 with $Re_\infty = 5 \times 10^6$ and $M_\infty = 0.15$. RANS simulations by SU2.

in the training database. For this reason a *relevance index* r_i is introduced:

$$r_i = \frac{\mathrm{d}V_i/\mathrm{d}\alpha}{\max(\mathrm{d}V_i/\mathrm{d}\alpha)} + \frac{d}{\max(d)} \tag{3.5}$$

where $\mathrm{d}V_i/\mathrm{d}\alpha$ are the derivatives of the latent variables with respect to the angle of attack α and d is the distance to the closest existing case in the parameter space. As showed earlier, the latent variables closely mimic the relationship between the aerodynamic force coefficient and the angle of attack. Therefore, the assumption made here is based on the premise that α serve as the primary control parameter in defining the relevance index. r_i is computed for each database airfoil and choose the highest value in the α-Re_∞ space. In this way a new case for each airfoil is selected after every refinement iteration.

A flow-chart of the procedure is reported in Figure 3.6.3. Various criteria can be employed to assess the quality of interpolation (termination condition in Figure 3.6.3). One meaningful

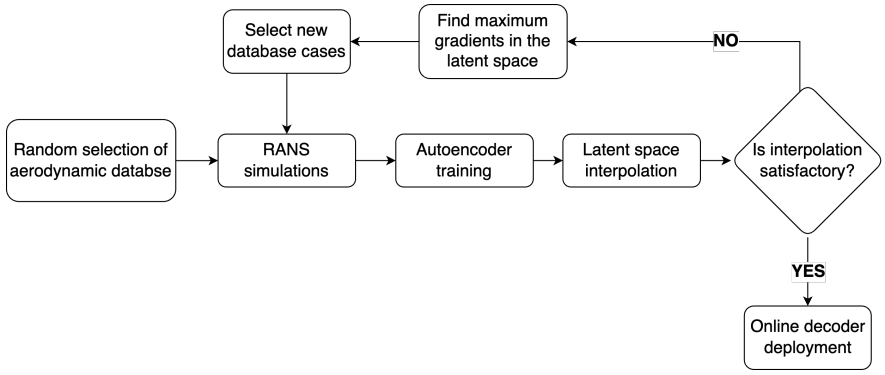


Figure 3.6.3: Flow-chart of the algorithm for the automatic database generation.

measure, offering insight into the convergence of the iterative procedure, is the prediction error on the testing dataset.

To test the capabilities of the algorithm for the adaptive database generation, the NACA 4420 is again considered. Figure 3.6.4 shows the latent variables as function of α for the NACA 4420 airfoil after 3 refinement iterations (+45 additional cases). Comparing this figure with 3.5.27,

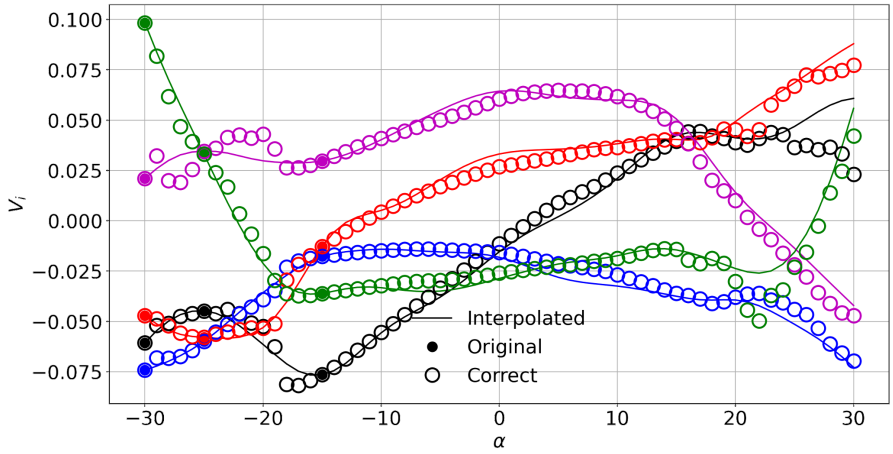
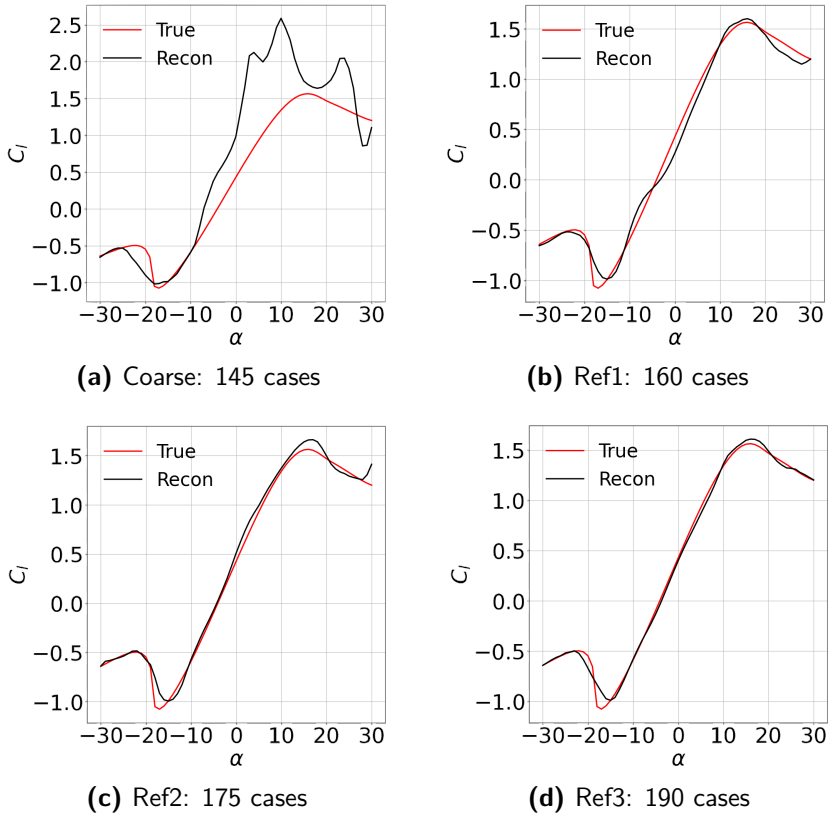


Figure 3.6.4: Latent variables as function of α for a NACA 4420. Comparison between the trained latent variables and the interpolated ones after 3 refinement iterations. –: V_1; –: V_2; –: V_3; –: V_4; –: V_5.

clearly, by refining the database, the interpolated latent variables follow the correct ones. This happens also if no additional cases were selected for the NACA 4420 during the refinement algorithm. More interesting the analysis of the predictive performance of the decoder. In Figure 3.6.5 the algorithm for the database refinement is evaluated, in terms of lift curves. The ground truth (obtained by RANS simulations) is compared with the lift curve computed from the flow fields generated by the decoder. In particular, from subfigure 3.6.5a to 3.6.5d the process begins

with the coarse database, followed by three iterations of the refinement algorithm. This result evidences the beneficial effect of refining the database only where it is necessary: with just few new cases there is a quick improvement of the results. This result is very promising, because, this fig-

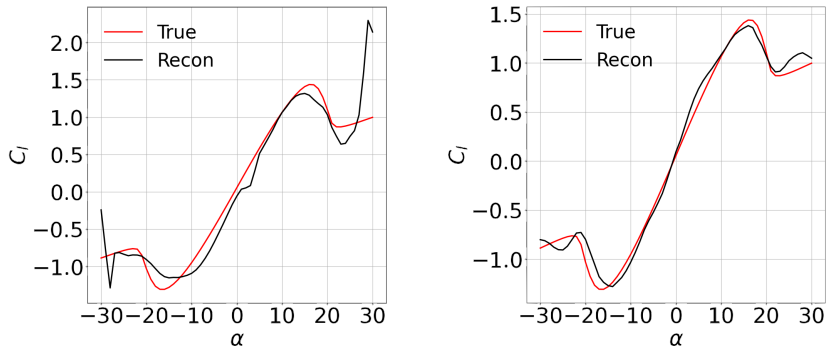


(a) Coarse: 145 cases **(b)** Ref1: 160 cases

(c) Ref2: 175 cases **(d)** Ref3: 190 cases

Figure 3.6.5: NACA 4420 lift curves, $M_\infty = 0.15$ and $Re_\infty = 7 \times 10^6$. Comparison between the ground truth (True) and the decoder output (Recon).

ure shows an integral information of 512 data points for each value of C_l, while the AE training procedure optimizes the differences between a data-set composed by $512 \times 256 \times 5 \times 190$ ($\sim$ 34 millions) data points. Where 512×256 are the cells of the computational grid; 5 are the input channels (the physical variables in input); 190 is the number of CFD simulations composing the final database. For this reason, the lift curves reported in Figure 3.6.5 and 3.6.6 represent only the 0.16% of the entire data generated by the AE, which was trained to optimize the error on a significantly larger data dimension. The differences between the ground truth and the predicted values appear in correspondence of the interpolation errors. Once again, it is confirmed that the algorithm here developed for database refinement accurately identified the appropriate cases for inclusion in the database. Regarding the whole AE output, Figure 3.6.7 shows a comparison between the flow fields obtained by the CFD (first rows) and the ones obtained by the



(a) Coarse: 145 cases

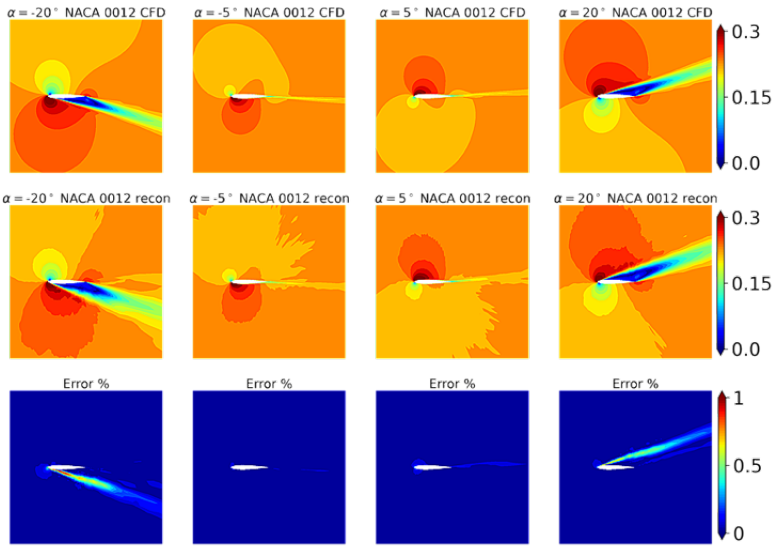
(b) Ref1: 160 cases

(c) Ref2: 175 cases

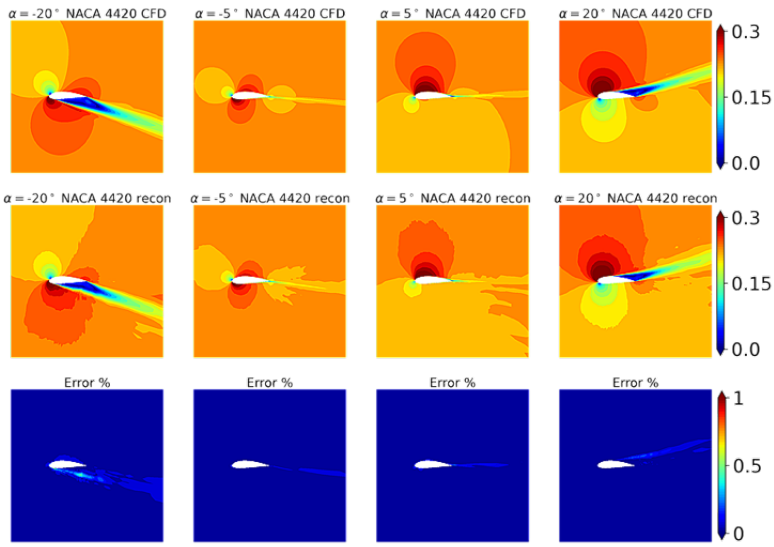
(d) Ref3: 190 cases

Figure 3.6.6: NACA 0012 lift curves, $M_\infty = 0.15$ and $Re_\infty = 5 \times 10^6$. Comparison between the ground truth (True) and the decoder output (Recon).

decoder, second rows (recon) in terms of local Mach number M. Third row is the percentage error between the reference RANS solution and the one predicted by the decoder. The error is computed with respect to the free-stream value of the Mach number $M_\infty = 0.15$. Note that the input variables during the training are ρ, p, u, v and $\hat{v}$; therefore, the contour maps of the local Mach number are obtained as composition of the predicted fields ρ, p, u and v.



(a) NACA 0012, $Re_\infty = 5 \times 10^6$



(b) NACA 4420, $Re_\infty = 7 \times 10^6$

Figure 3.6.7: Comparison between the ground truth (CFD) and the decoder output (Recon). The errors are computed as percentage of the free-stream value for the Mach number $M_\infty = 0.15$.

The errors are coherent with the interpolation error. The reconstructions are not smooth even thought the errors are small. The smoothness of the solution can be obtained by applying a filtering; by imposing constrains in the loss function during the training phase or using a wider training data-set. At this stage, the decision was made to retain the results obtained using a simple AE to assess its performance.

After assessed the possibility to predict new unseen flow fields around airfoils present in the database, even for a limited range of angles of attack, the AE was tested on a new airfoil: the NACA 2412 at $Re_\infty = 5 \times 10^6$. Figure 3.6.8 shows that the interpolated latent variables are in

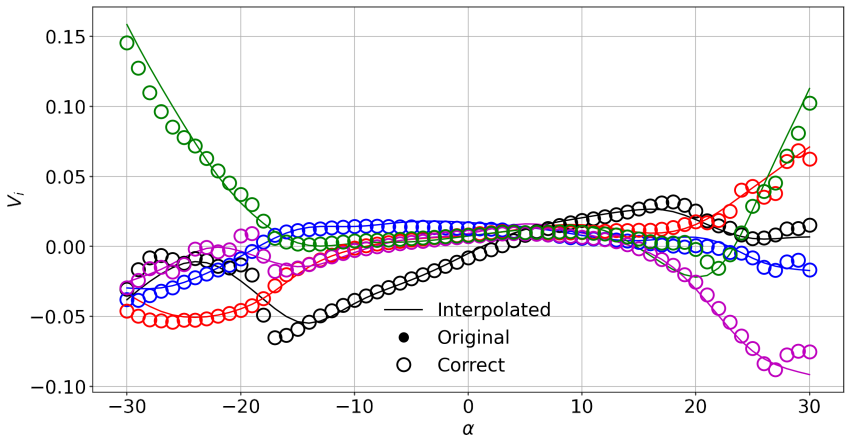


Figure 3.6.8: Latent variables as function of α for a NACA 2412. Comparison between the interpolated latent variables and the correct ones. −: V_1; −: V_2; −: V_3; −: V_4; −: V_5.

agreement with the ones obtained by the encoder, even though the NACA 2412 is not present in the training data-set (notice that no filled dot is present in the figure). The flow predicted by the decoder (Figure 3.6.9 and 3.6.10) are in good agreement with the CFD simulation as already seen for the other airfoils. Figure 3.6.9 shows the lift curve predicted by the decoder, the errors with respect the one obtained with RANS, correspond to the interpolation errors visible in Figure 3.6.8.

It is important to notice that the solutions obtained by the decoder are instantaneous: it needs fractions of second to retrieve a new unseen RANS solution. The computational time for generating a new flow field with the decoder is $\sim 1.25 \times 10^5$ times less than performing a CFD simulation.

Moreover, an interesting feature of such technologies is the portability: the storage needed for the decoder is 29 Mb, while, for each RANS simulation is 13 Mb (in binary Paraview format), or 1.1 Mb (in python numpy format, adopted for the training). In the best case, it needs 209 Mb to store the training database. However, to predict every flow field of every NACA airfoil for Reynolds numbers $o(10^6)$, only the decoder is needed. To prove the potential of this technology, th emodel was ported in an Android smartphone, which can now predict the flow

Figure 3.6.9: Lift curve for the NACA 2412 at $Re_\infty = 5 \times 10^6$. Comparison between the ground truth (True) and the decoder output (Recon).

field around airfoils in terms of primitive variables and turbulence quantities in fractions of second.

The results obtained with the algorithm for the adaptive generation of the database, are very promising for the development of deep learning ROMs when the computational cost for the database generation is a main bottleneck or the physics of the phenomenon is not completely known. The time reduction for the flow field generation and the storage saving (portability) with such models will drastically scale when moving to high-fidelity databases such as LES and DNS.

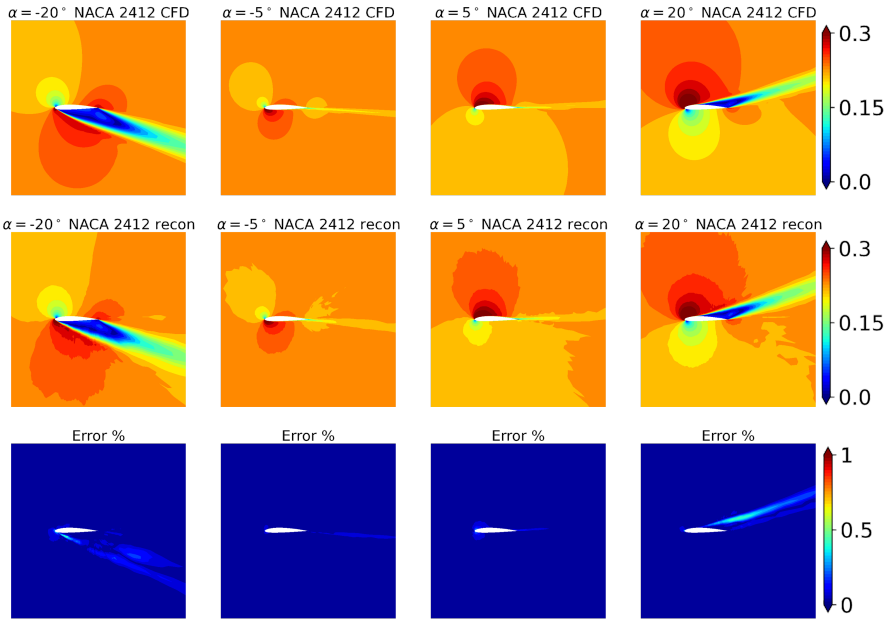


Figure 3.6.10: Comparison between the ground truth (CFD) and the decoder output (Recon). The percentage error is computed with respect to the free-stream value for the Mach number.

3.7 Unsteady aerodynamics

The results obtained for the static stall are very promising for advancing the investigation into the unsteady regime. In this phase, the focus shifts towards understanding the dynamic stall of airfoils subjected to periodic pitching motion. Dynamic stall introduces a level of complexity significantly greater than that of static stall. This phenomenon is widely investigated in literature, in particular by EFD [88, 89], CFD [90, 91] and semi-empirical models [92].

The selected database is composed by the same airfoils adopted in database **DS2** of section 3.2.1. Figure 3.7.1 shows the lift curves for symmetric NACA airfoils for the static stall and the dynamic stall. Regarding the static stall (Figure 3.7.1a), as the angle of attack increases, the phenomenon exhibits a linear behavior at the initial stages, specifically for small angles of attack. However, as the angle of attack continues to increase, lift coefficient reaches a maximum value where stall occurs. At this point, the phenomenon transitions into a strongly non-linear regime. Figure 3.7.1b shows the lift curves of the same airfoils with a periodic pitching motion in dynamic stall regime. One initial observation is the presence of a hysteresis cycle. In this scenario, when the angle of attack is increased, the lift coefficient exhibits a continuous increase even beyond the static stall angle of attack, to the extent that it displays a spike as it approaches the stall. This distinctly non-linear behavior is attributed to the detachment of vortices from the airfoil's leading edge. Conversely, when decreasing the angle of attack, the airfoil fails to re-

gain lift until flow reattachment occurs. These results were obtained by SU2 RANS/URANS solver with the SA turbulence model. The URANS simulation of the 4 airfoils were conducted at

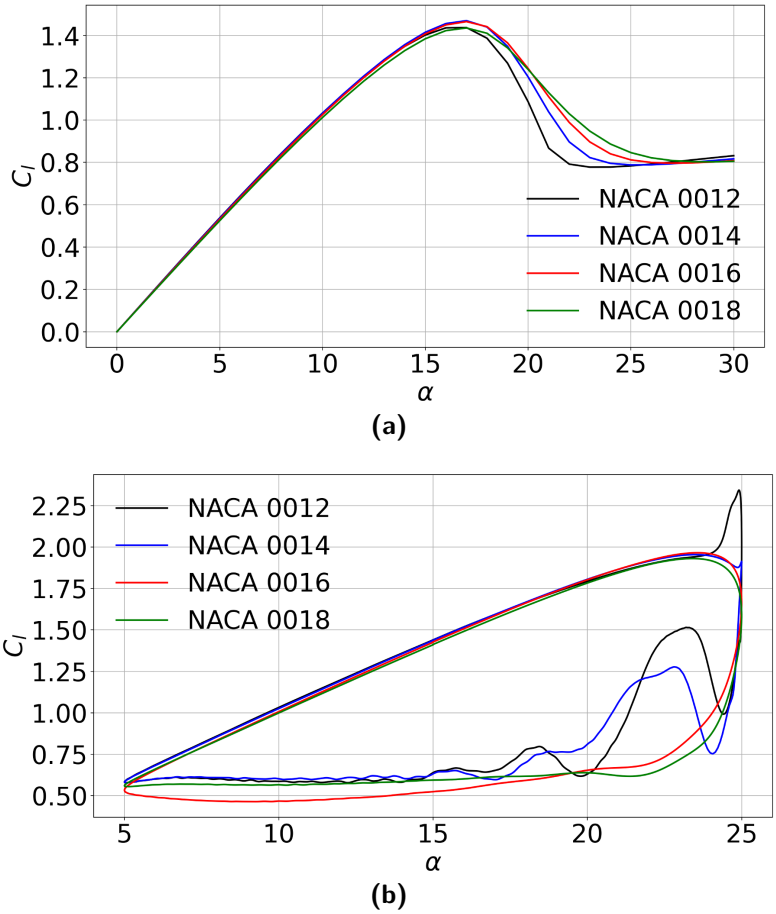


Figure 3.7.1: Lift curves for NACA symmetric airfoils. **(a)** Static stall, RANS simulation, $M_\infty = 0.15$, $Re_\infty = 5 \times 10^6$; **(b)** URANS simulation, $M_\infty = 0.15$, $Re_\infty = 5 \times 10^6$, $k = 0.1$.

$Re_\infty = 5 \times 10^6, M_\infty = 0.15$. The angle of attack follows a periodic pitching motion around the 25% of the airfoil chord as follows:

$$\alpha = \alpha_m + \alpha_0 \sin\left(2\bar{f}t\right) \tag{3.6}$$

where α_m is the mean angle of attack, α_0 is the amplitude of the oscillation, $\bar{f}$ is the reduced frequency and t is the time. In the present database $\alpha_m = 15°$, $\alpha_0 = 10°$ and $k = 0.1$. The reduced frequency $\bar{f}$ is defined as

$$\bar{f} = \frac{\omega c}{2V_r} \tag{3.7}$$

where ω is the angular frequency, c is the airfoil chord and V_r is a reference velocity (the free-stream velocity in this case).

Within the database, two parameters are subject to variation: the airfoil thickness and the angle of attack (which can be regarded as a temporal parameter according to eq. (3.6)). Nevertheless, the complexity of this database is further increased by **changes in the underlying physics** of the dynamic stall phenomenon. Upon examination of Figure 3.7.1b, the lift curves corresponding to the NACA 0012 and 0014 airfoils exhibit a spike at the stall and strong oscillations in the deep stall region. In contrast, such behavior is absent in the case of the NACA 0016 and 0018 airfoils, where the lift curves exhibit a smoother profile. This difference is explained investigating the flow around the airfoils in the two scenarios. Specifically, the NACA 0012 and 0014 airfoils experience the separation of a leading-edge vortex, known as the dynamic stall vortex (DSV). In contrast, for the NACA 0016 and 0018 airfoils, the DSV has no time to separate from the airfoil surface. The database is composed by 1800 randomized snapshots (450 for each airfoil), where the 15% is kept for evaluation.

The AE architecture adopted for the analysis of the *unsteady* database is the same already described for the database **DS2** of section 3.2.1. Also in this case, the regularization is applied with a penalization of the L_2 norm of the encoder weights.

Various training of the AE were performed varying the latent space dimension. Figure 3.7.2 shows the training history for $N_{ls} \in [1, 5]$. Using $N_{ls} = 1$ and $N_{ls} = 2$, it is evident that convergence is not attained. However, convergence is achieved when $N_{ls} = 3$, and with $N_{ls} \geq 3$ there is no significant enhancement in training performance. It is pertinent to highlight that, in this context, a difference from the prior approach is observed. Specifically, the use of latent variables equal to the number of database parameters is no longer sufficient; an additional latent variable is necessitated. This variation can likely be attributed to the heightened complexity present in the dataset, characterized by an additional factor: the variation of the underlying physics associated with the stall phenomenon between thin and thick airfoils.

The projections of the latent space thus obtained (for $N_{ls} = 3$) are reported in Figure 3.7.3. Two aspects of the latent space require investigation: sorting and shape. It is sorted with respect the two database parameters: time and airfoil thickness. In particular, Figure 3.7.3b shows the time evolution of the latent space. Regarding the shape, the latent space also captures the periodicity of the phenomenon: the latent variables, associated to an airfoil, follow a closed curve. Moreover, it is possible to notice a different behaviour of the latent variables for the NACA 0016 and 0018 with respect to the NACA 0012 and 0014. This difference reflects the change in the physics of the stall phenomenon discussed above.

For a more in-depth understanding of this phenomena, it can be useful to examine the latent variables as function to the angle of attack. Figure 3.7.4 shows the three latent variables for the four airfoils plotted as function of a. In all the latent variables, one can observe the change in stall physics, aligning with the phenomenon discussed before: a spike at the stall and strong oscillations for the NACA 0012 and 0014, while the NACA 0016 and 0018 exhibit smoother

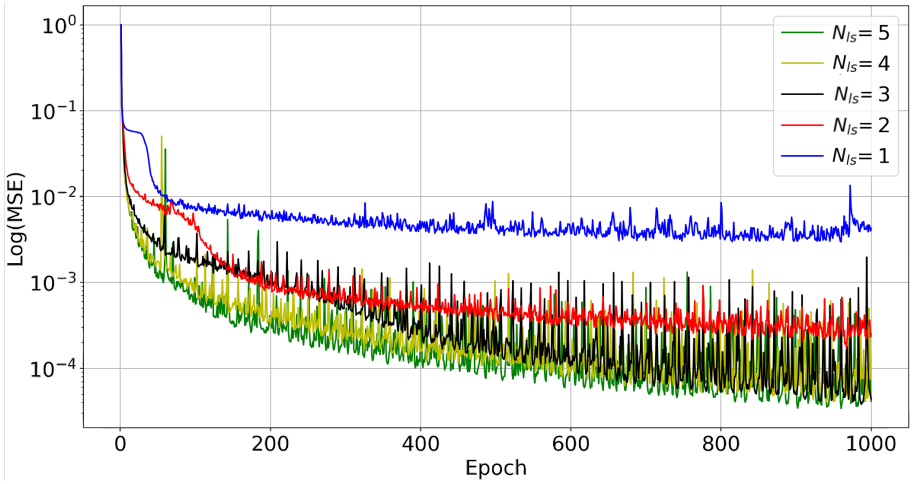


Figure 3.7.2: Loss function as function of the epochs. AE trained on the unsteady aerodynamic database. Comparison for $N_{ls} \in [1,\ 5]$.

curves.

At this point, sufficient insights have been gathered regarding the physical characteristics embedded in the latent space. An evaluation of the correlation between the latent variables and the aerodynamic coefficients is warranted to gain a comprehensive understanding of how to make practical use of these findings. Performing a non-linear regression, with a 3rd order polynomial of the latent variables on the aerodynamic coefficients, leads to accurate representation of the aerodynamic force and moment coefficients. Figure 3.7.5 shows this results of the regression for the NACA 0014 airfoil. Figure 3.7.6 shows the same result for all the airfoils in the database. The regression conducted here can be regarded as a ROM for the lift, drag, and aerodynamic moment coefficients. This result clearly indicates the presence of a non-linear correlation between the latent variables and the aerodynamic force and moment. Given that the latent variables effectively capture changes in the underlying physical phenomenon, this ROM exhibits the same capability.

It is crucial to investigate the prediction performance of the trained decoder, particularly in the generation of new, unseen flow fields.

A first step is to investigate the capabilities of the AE to generate new fields in the dataset interpolation region. The latent space mapping on the database parameters (thickness and time) is showed in Figure 3.7.7, where the latent variables for a NACA 0013 airfoil are extracted. Feeding the interpolated latent variables to the decoder, it provides in output an accurate the time evolution of the flow field for a NACA 0013 airfoil in dynamic stall regime. Some of the snapshot produced by the decoder are reported in Figure 3.7.8 for the most important phases of the phenomenon. Figures 3.7.8a and 3.7.8b are in the upstroke phase, when the flow is still attached to the airfoil surface. 3.7.8c and 3.7.8d are in the downstroke part of the oscillation. The former is

at the stall, where the DSV vortex is leaving the airfoil surface (visible in the local Mach number contour and in the bump present in the surface pressure coefficient); the latter is in the deep stall region, where the flow is slowly reattaching on the airfoil surface. The new *synthetic* flow field is reliable and in good agreement with the URANS simulation of the NACA 0013.

At this point, it is worth to investigate the extrapolation capabilities of the AE. For this reason, the latent variables corresponding to a NACA 0020 airfoil are extracted from the latent space mappings. Feeding these variables to the decoder, the result is again a reliable flow filed in agreement with the URANS solution. Snapshots of the predicted field and URANS simulation are reported in Figure 3.7.9. It is possible to notice the different behaviour in the dynamic stall evolution between the NACA 0013 and NACA 0020 airfoils. The change in the stall physics, as previously discussed, is effectively captured by the AE. Such results show how an AE can *understand* the underlying physics of a phenomenon from the input data, even for unsteady and strongly non-linear phenomena.

At this stage it is worth to analyze the computational cost of the AE, and to compare it with the CFD simulations. Table 3.7.1 summarize the time for obtaining a converged URANS so-

	CPU Time (s)
URANS (CPU)	$\sim 1.33 \times 10^6$
Decoder (CPU)	~ 8.06
Decoder (GPU)	~ 0.080

Table 3.7.1: Computational cost to obtain the flow field around a NACA 0012 airfoil with an unsteady pitching motion at $Re_\infty = 5 \times 10^6$, $M_\infty = 0.15$, $\bar{f} = 0.1$. Mesh size: 131072 elements, 9000 timesteps per cycle. URANS simulation performed with SU2. CPU: Intel Xeon, 2.30GHz; GPU: GeForce GTX Titan X.

lution with SU2 with the time needed to generate the flow field using the AE. The time for the decoder prediction drops with a $\sim 1.6 \times 10^5$ factor with respect to the CFD simulation. By executing the decoder on a GPU, it is possible to improve the speed of another 100 factor. The time for training the AE with 850 epochs is $\sim 1.8 \times 10^4$ seconds (2 orders lower than a single URANS simulation).

the comparison is drawn with URANS simulations. It should be noted that when considering a LES or DNS database, the computational acceleration becomes significantly more pronounced.

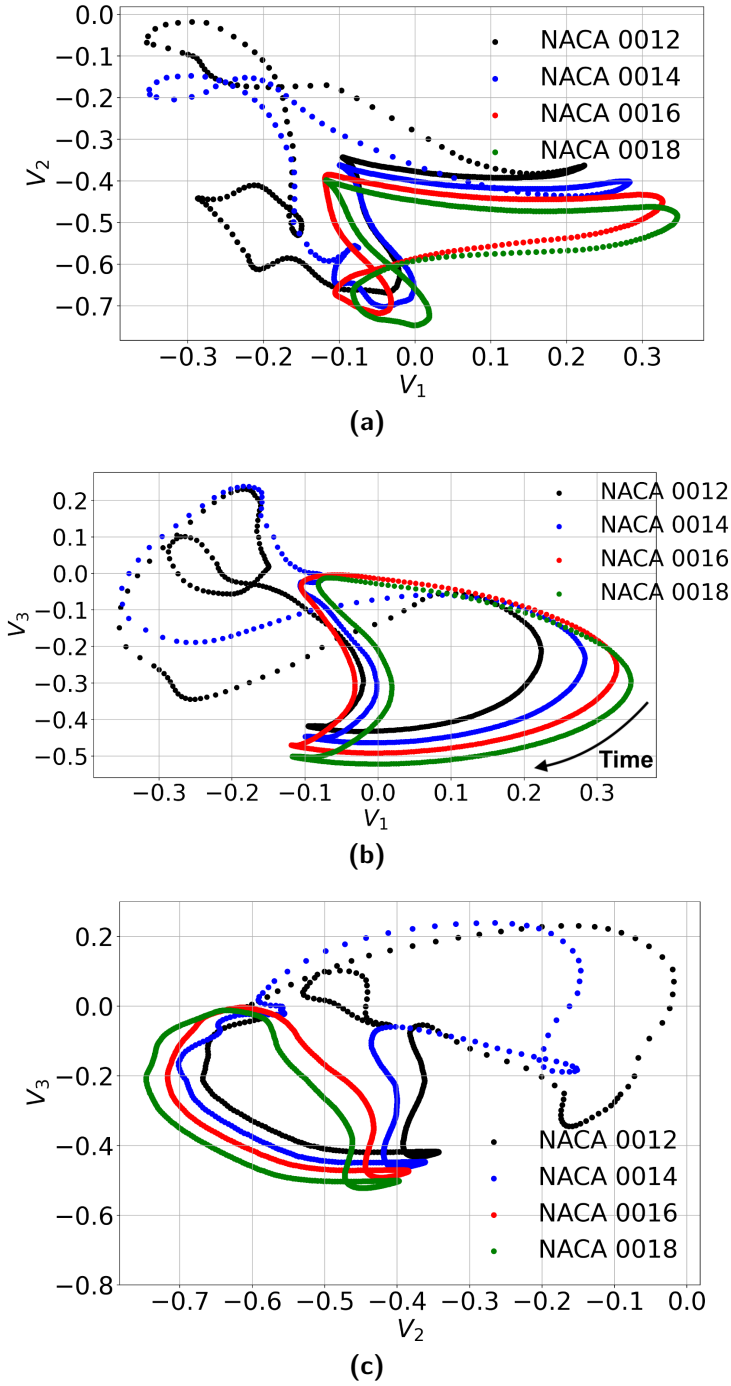


Figure 3.7.3: Latent space projections of the AE trained on the unsteady aerodynamic database.

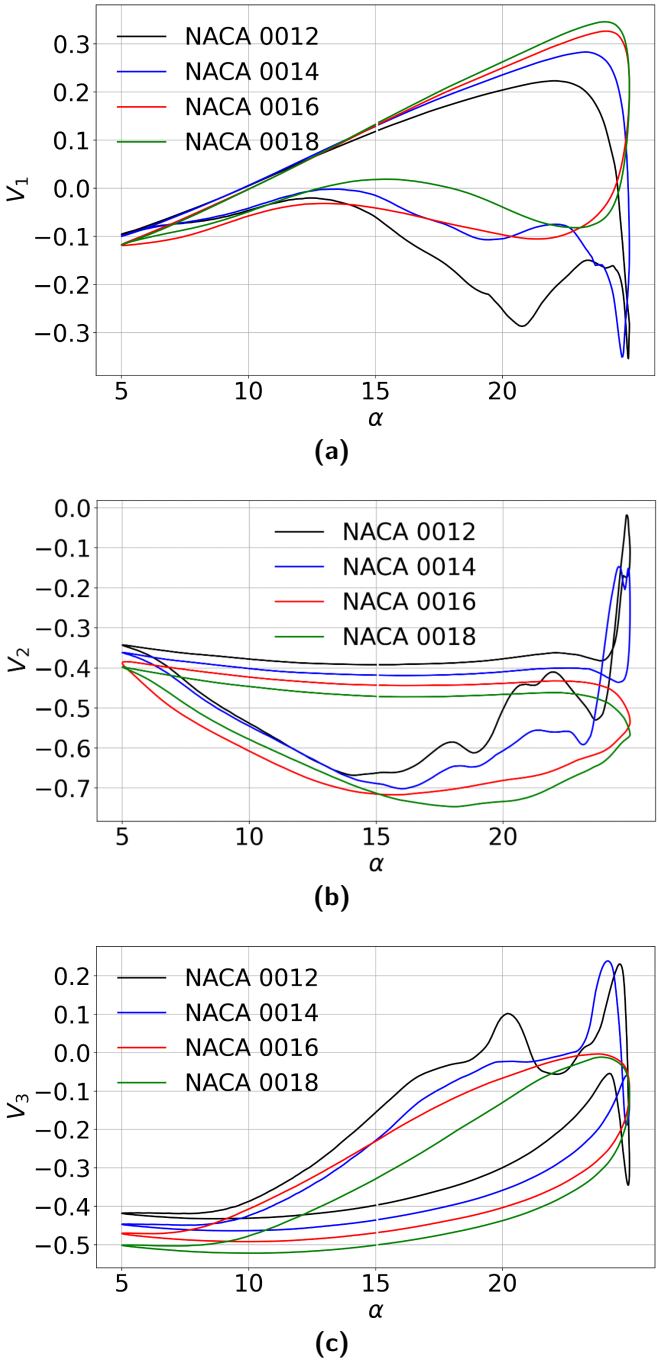


Figure 3.7.4: Latent variables as function of α for the unsteady aerodynamic database.

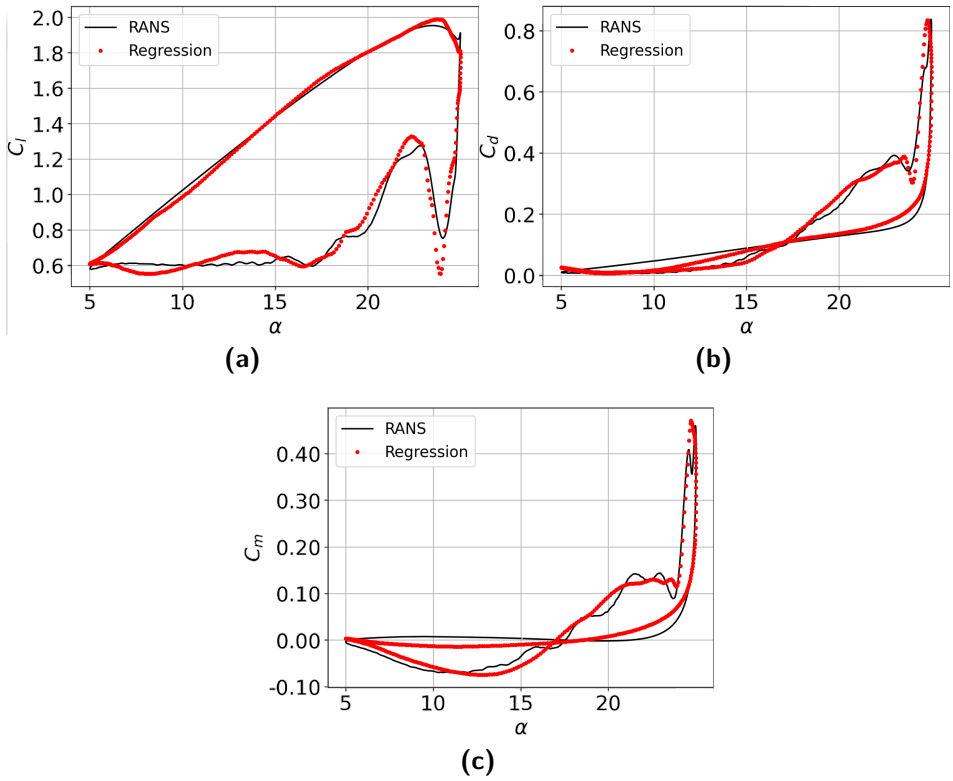


Figure 3.7.5: Aerodynamic force and moment coefficients compared with the non-linear regression of the latent variables. NACA 0014, URANS simulation, $Re_\infty = 5 \times 10^6$, $M_\infty = 0.15$, $\bar{f} = 0.1$. **(a)**: lift coefficient; **(b)**: drag coefficient; **(c)**: moment coefficient with respect $1/4$ of chord length.

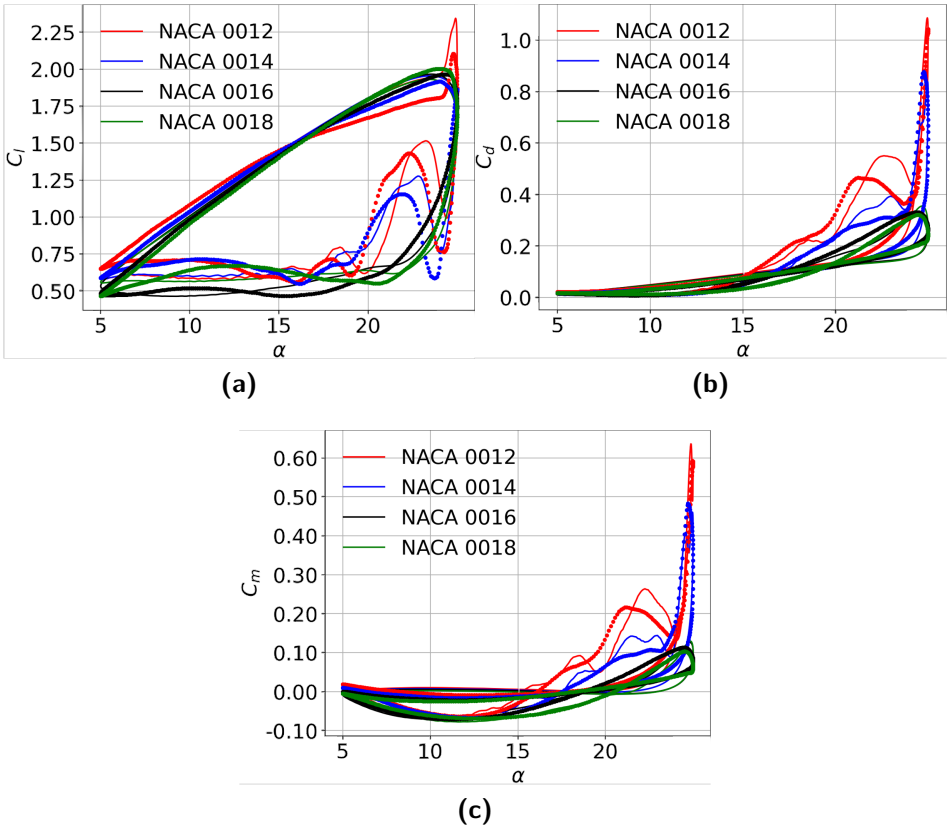
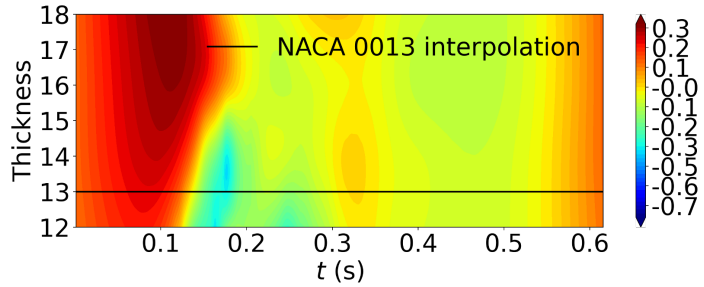
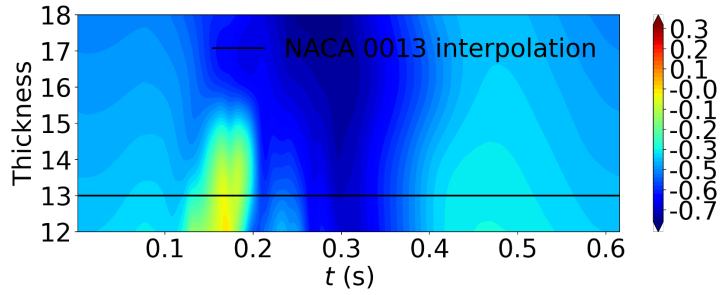


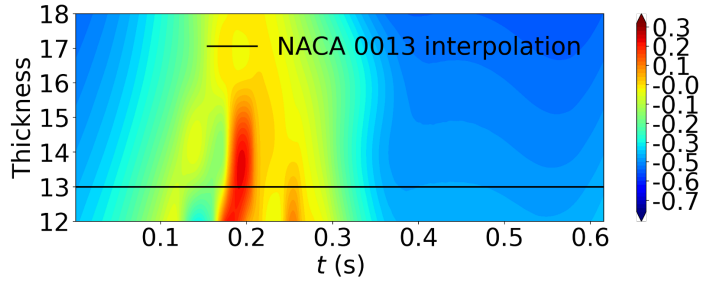
Figure 3.7.6: Aerodynamic force and moment coefficients compared with the non-linear regression of the latent variables. Continuous lines obtained with the URANS simulations; dots obtained from the non-linear regression of the latent variables. URANS simulations, $Re_\infty = 5 \times 10^6$, $M_\infty = 0.15$, $\bar{f} = 0.1$. **(a)**: lift coefficient; **(b)**: drag coefficient; **(c)**: moment coefficient with respect $1/4$ of chord length.



(a) V_1



(b) V_2



(c) V_3

Figure 3.7.7: Latent space mapping on the database parameters (airfoil thickness and time). Unsteady aerodynamic database.

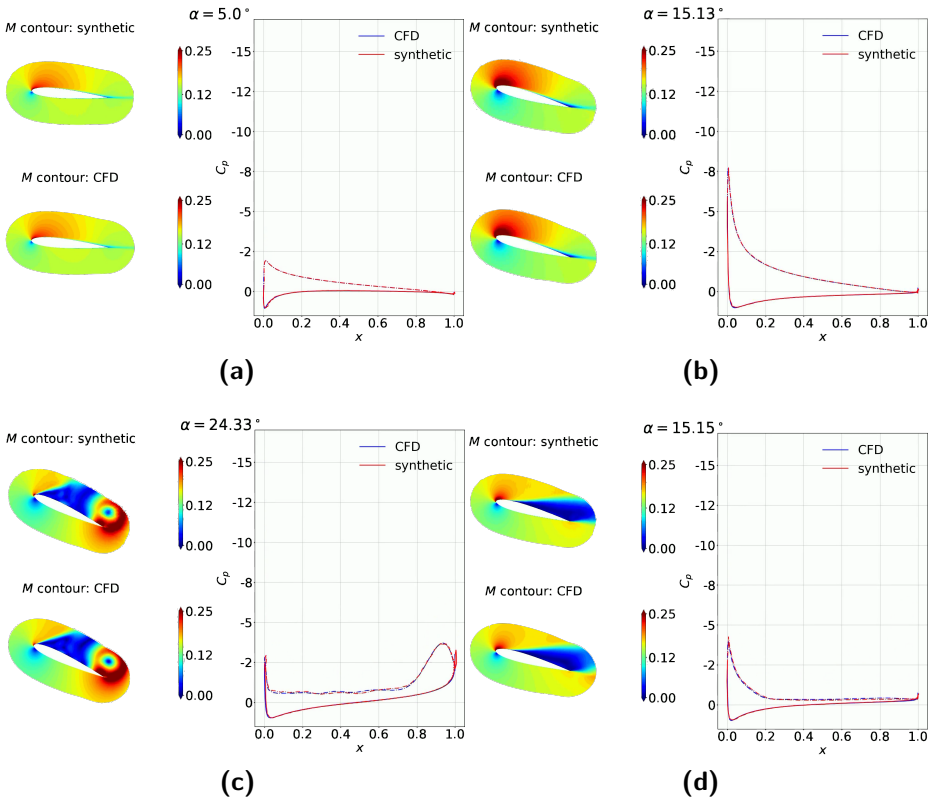


Figure 3.7.8: Snapshots of the URANS AE prediction (*synthetic*) for a NACA 0013 airfoil, compared with URANS CFD simulation (*CFD*) at $Re_\infty = 5 \times 10^6$, $M_\infty = 0.15$, $\bar{f} = 0.1$. Local Mach number contour and surface pressure coefficient.

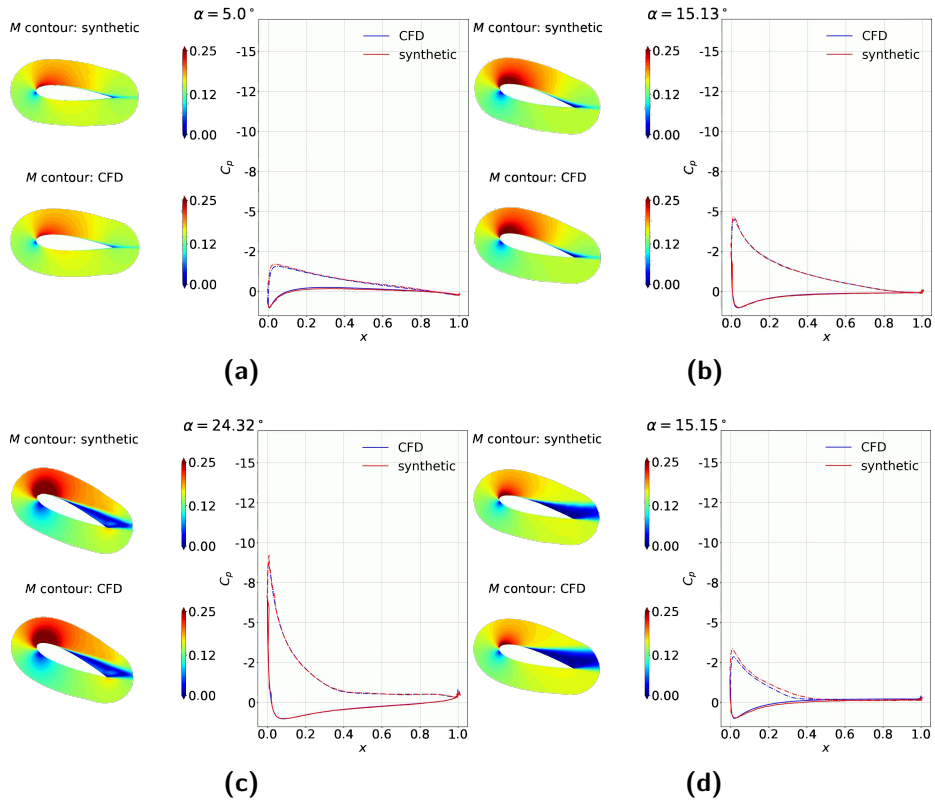


Figure 3.7.9: Snapshots of the URANS AE prediction (*synthetic*) for a NACA 0020 airfoil, compared with URANS CFD simulation (*CFD*) at $Re_\infty = 5 \times 10^6$, $M_\infty = 0.15$, $\bar{f} = 0.1$. Local Mach number contour and surface pressure coefficient.

4

Uncertainty Quantification for Autoencoders

As ML and specifically AE techniques, continue to gain popularity, it is important to assess the confidence level that can be assigned to the generated results. Uncertainty Quantification (UQ) has received considerable attention in computational science especially in reference to the effect of variability present in the system and limited precision in the definition of the operating conditions of interest; probabilistic approaches in which the inputs are represented in terms of random variables are well established and provide a rigorous framework to characterizing the confidence (with a frequentist perspective) in the results. Another potential source of uncertainty is the introduction of modeling assumptions, for example choices in the design of the computational approach used or assumptions introduced in physical models. This is sometime referred to model form uncertainty and is more difficult to represent although in many cases critical to establish confidence in the predictions.

It is useful to compare and contrast the differences between uncertainties in a well understood *classical* computational model vs. a ML approach. The selected context is to study airfoil aerodynamics [84]. This is a traditional problem in Fluid Mechanics and it has been approached using numerical methodologies since computers were first introduced many decades ago. The *conventional* prediction strategy is based on the solution of governing equations representing transport of mass and momentum (assuming an isothermal environment), the Navier-Stokes equations. In most industrial applications, a modified version of the equations, which represent statistically the effect of turbulent fluctuations on the steady state behavior of the flow is used;

this approach solves the Reynolds Averaged Navier-Stokes (RANS) equations [6]. Although the focus of the discussion and the results in this work is on RANS predictions, extensions to other physical problems or more complex geometrical configurations would be straightforward from the UQ perspective.

To illustrate the difference between a conventional RANS modeling approach and a ML-based methodology, start by referring to Figure 4.0.1. One refer to the two approaches as *equation-driven* and *data-driven* respectively. In both scenarios the objective is to predict some relevant quantity of interest, such as the lift and drag of the airfoil, given a sufficient specification of the physical problem: in this case, the inputs are the parameters describing the geometry and the operating conditions. In an equation-driven approach, the computational model is constructed

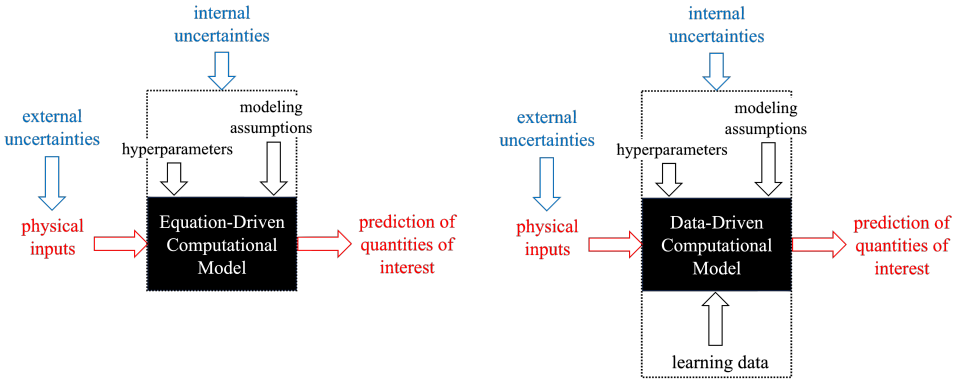


Figure 4.0.1: Schematic representation of a conventional computational model (equation-driven) vs. an ML approach (data-driven)

using a discretization of the governing equations - which typically includes a mesh, a scheme to approximate the differential terms in the equations - and a solver for the resulting algebraic equations. Both the discretization and the solution steps include user-specified parameters that control the size of the mesh, the order of accuracy of the method, etc.; these quantities (in most cases a small number of parameters) are referred to as hyperparameters. In addition, a series of assumptions that define physical closures and approximations for the problem must be defined, for example the description of the fluid in terms of viscosity and an equation of state. It is fairly obvious that the predictions confidence is a direct consequence of uncertainties present in the hyperparameters and the modeling assumptions (namely internal uncertianty) and the physical inputs (external uncertainties).

In a data-driven approach the overall picture and the problem statement is similar (Figure 4.0.1): given the physical input, predict the quantities of interest; however, there are remarkable differences. The first and most important is the need to supply data to carry out the learning phase, i.e. to *train* the model; this is typically executed out off-line and involves an optimization process. The final result is essentially a function that maps the physical inputs into the prediction of the quantity of interest (online prediction). This function, and the overall optimization

process is affected by a potentially large number of hyperparameters which control the learning speed, the cost function (loss function in the ML jargon) etc. In addition, and perhaps more importantly, the dataset used for the learning phase impact directly the prediction ability of the model. The learning data plays the same role as the governing equations and even if not effectively used in the actual prediction phase, both the equation and the data are fundamentally part of the computational model. Finally, the mapping function is a non-linear transformation defined as a network with layers and connections that effectively represent the model form. In ML approaches, this mapping is defined by a very large number of adjustable parameters (weights) are evaluated as part of the optimization process. From an uncertainty quantification perspective, the addition of the learning phase is the main new element to consider and adds to the amount of internal uncertainty.

In the computational experiments reported below, RANS solutions obtained using the equation-driven approach will be considered as inputs to the AEs; this will enable to carefully control the characteristics of the learning dataset and to have easy access to ground truth to assess the AE predictions and their uncertainty. The investigation entails assessing the impact of different sources of uncertainty on the predictions, as well as the influence of additional variability in specifying the operating conditions. This will enable to predict the total uncertainty and its dependence on the prediction scenario selected. Lastly, a comparison is made between the current predictions and a Gaussian Process Regression approach to illustrate the differences and the superior ability of AEs to identify high- and low-confidence in the predictions scenarios in the absence of ground truth data.

4.1 Previous work on uncertainty analysis in machine learning

Recently, there has been a growing focus on investigating and quantifying uncertainties in ML. Notably, Abdat et al. in [93], Kabir et al. in [94] and Psaros et al. in [95] have provided insightful reviews of UQ methods for deep learning techniques and comprehensive overviews of the prevalent challenges. These reviews classify UQ methods in ML into two main categories: *Bayesian techniques* (such as Monte Carlo dropout, Markov chain Monte Carlo, and VAEs [96]) and *Ensemble techniques* (including random weight initialization [97] and snapshot ensembles [98]).

Ensemble methods have demonstrated exceptional performance in a few studies, and are becoming more widely adopted; however, it is not always clear how the performance and in particular, confidence measure on the predictions are related to different sources of uncertainty. Evaluating the performance of UQ methods for ML is complicated due to the often unknown data-generating distribution. Predictions (and their uncertainty) varies based on the quality and quantity of measurements, the model architecture and the hyperparameters. The assessment of the robustness of ML models in the presence of a distribution shift, where model inputs diverge from the training data distribution, presents an ongoing challenge. In this context it is crucial

to develop UQ approaches that can identify such situations and explicitly provide predictions with low confidence. In other words, UQ is essential for the safe integration of ML in real-world engineering applications, thus clarifying inherent accuracy limitations both in interpolatory and extrapolatory settings.

UQ for AEs remains an area with limited coverage in the literature. The most common approach involves the use of VAEs (e.g. [99]). In this approach the latent variable embedding is augmented with a stochastic representation often described in terms of Gaussian random variables. More recently, Yong and Brintrup in [100], or Perini et al. in [101], employed Bayesian AEs for UQ in the context of anomaly detection. After training, these models produce stochastic predictions by sampling the latent variables. Again in this case, it is unclear how to quantify the effect of individual sources of uncertainty and how to distinguish between internal and external variability.

The present work falls within the ensemble framework, which has been extensively adopted and discussed in the context of deep neural networks, as highlighted by Pickering et al. in [102]. Numerous studies, including those by Hansen in [103] and Lakshminarayanan et al. in [97], have also explored the potential of ensembles to provide robust UQ for deep learning models.

Multiple ensembles are constructed by explicitly considering a diverse set of internal uncertainties (training data, hyperparameters, architecture) and enabling the representation of external uncertainties both in interpolatory and extrapolatory settings. Although the results focus on predictions of aerodynamic performance of an airfoil, the findings provide useful general insights into the AE performance.

4.2 Design of the input dataset

The performance of data-driven approaches, such as AEs, is critically dependent on the data used in the learning phase. In this research, a dataset was devised with parameterization involving two variables, facilitating the systematic construction of a process for controlled coverage of the solution space. This enables to characterize the effect of using ML for tasks involving either *interpolation* or *extrapolation*.

The input data typically used in convolutional autoencoders consist of images of fluid flow characteristics of wing sections when the angle of attack (a) and the airfoil thickness (t) are changed. Specifically, symmetric NACA airfoils are adopted [84] with $t \in [4\%,\ 20\%]$ of the chord length and angles of attack $a \in [-30°,\ 30°]$.

The flow solutions are described in terms of the pressure p and the velocity components u and v computed using a structured grid (see Figure 3.2.3a). The computations are carried out in two-dimensions and under steady flow conditions using the open-source CFD solver SU2 [44] and adopting the $k - \omega$ SST Reynolds-Averaged Navier-Stokes closure. The free-stream Reynolds number (Re_∞) and Mach number (M_∞) are fixed for all the simulations: $Re_\infty = 6 \times 10^6$ and $M_\infty = 0.15$.

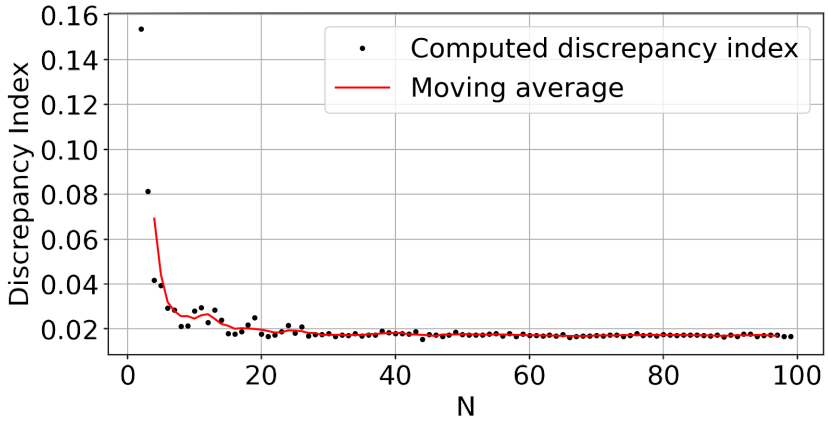


Figure 4.2.1: Discrepancy index as function of the number of input cases. The cases are selected using the Latin Hypercube Sampling algorithm.

As mentioned earlier, in this work convolutional AE are employed. In section 3.2.1, it was investigated training carried out using images (i.e. RGB values) and raw data which instead represent the values of the physical variables. The latter showed improved performance and have been adopted here. Related activities in the literature [51] involving datasets of airfoil flows either a voxelization of the geometry or a masked representation based on signed distance function has been used to distinguish between the region inside and the outside the airfoil. These approaches introduce approximations that preclude a clear assessment of the accuracy of the AE prediction. In the present work, an alternative approach is used, that is accurate and efficient and precisely represents the flow field in a Cartesian domain. As already anticipated in section 3.2, the input images are obtained by considering a transformation of the physical space into a notional space defined by the structured grid used in the computation. In Figure 3.2.3 the curvilinear mesh is mapped in the index space *i-j*. This transformation also emphasizes regions close to the walls in which the boundary layers properties vary very rapidly; the grid clustering required by the RANS simulations is effectively eliminated in the *i-j* space and the corresponding fields are smooth and well represented. Needless to say the correspondence between the physical space and the notional space is easily invertible.

The input dataset for the AE learning phase is a collection of RANS solutions with different values of angle of attack a and airfoil thickness t; each solution is arranged in the form of a tensor with dimensions $n = n_i \times n_j \times c$ where n_i and n_j corresponds to the dimensions of the computational grid (512×256) while c is the number of physical variables (in this case 3). Note that the airfoil geometry and the controlling parameters a and t are not given explicitly as input to the AE: they are not required to complete the learning phase. The relationship between the latent variables and these parameters, particularly when employing the decoder as a generative model, will be explored later. The number of input cases N and the selection of the corresponding parameters (a and t) can be carried out in different ways. The most general approach is to

randomly sample a and t without repetitions. It is well know that this might lead to clustering thus reducing the effectiveness of the learning. One strategy commonly adopted is to define batches as a subset of the input cases (picking $N_b < N$ input cases at the time) and perform the learning optimization iteratively; the batches can be selected to maximize diversity between the cases [104]. In this work, the process is simplified by employing a traditional quasi-random strategy, namely Latin Hypercube Sampling (LHS) [86] that ensures sparsity and coverage of the space of the input parameters. The batches can be constructed by selecting randomly among the cases because diversity is already guaranteed. As an example Figure 4.2.1 shows the discrepancy index [87] (Centered Discrepancy) as a function of the number of input cases considered N. It is clear that considering more than 50 cases would not provide additional diversity in the dataset and therefore should not improve the learning phase; therefore $N = 44$ was selected. The corresponding input parameters a and t are reported in Figure 4.2.2. As is common practice in ML approaches, the input set is split into a *training set* that is used to carry out the optimization process and select the AE parameters, and a *validation set* that is used to verify the accuracy of the resulting model. In this work a 85%-15% split between training and validation cases is adopted.

Intentionally, in this work, the adopted input dataset is small. This is intended to duplicate realistic engineering applications in which it is impractical to generate and manage an extensive number of input cases. In section 4.5.2 the effect of different sizes of the input database is investigated, by considering two additional tests with $N = 33$ and $N = 59$ cases. The results indicate no noticeable differences, confirming that the information contained in the original database has a sufficient degree of diversity and coverage of the parameter range of interest, as shown in Figure 4.2.1. One of the main reasons for carefully constructing the input database is to assess the ability of the AE to operate in diverse regimes of data availability. In this case, in addition to a classical *extrapolatory* task (for $a > 30°$) a challenging *interpolatory* task for $a \in [-15°, 0°[$ was designed. For the latter, no data is present in the learning set, but given the symmetry of the airfoil geometry, it is known that the solution characteristics are mirrored from positive angles of attack. Are the errors associated with the extrapolatory task always more pronounced that those related to the interpolatory task? Is it possible to effectively characterize the uncertainty in the predictions in both of these scenarios? These are some of the questions that are explored and addressed in the results section.

4.3 Hyperparameter tuning

ML approaches are characterized by the presence of a large number of parameters that, on one hand provide flexibility in performing various prediction tasks but, on the other, introduce a burden for the user and create challenges for reproducibility. The vast majority of the parameters are obtained as part of the learning phase, as a solution to the optimization process. In the present context, this corresponds to the search for the *optimal* choice of ℓ_e and ℓ_d. However,

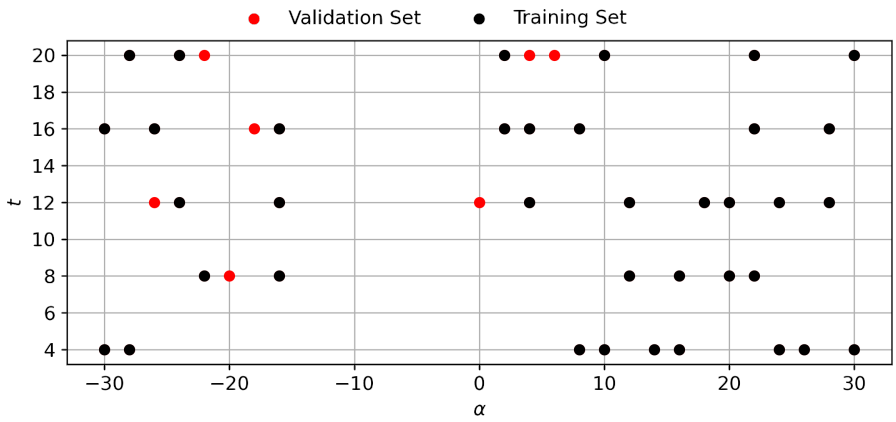


Figure 4.2.2: Parameters corresponding to the input cases selected using LHS. A $85\% - 15\%$ random split between training and validation cases is used. Note that, typically, an ensemble of (50) models is constructed, and therefore the split between test and training set reported in the figure corresponds to one member of the ensemble.

an additional set of free parameters (namely the hyperparameters) has been introduced in the previous sections, specifically the learning rate, the batch size (N_b) and the regularization coefficient (β).

Rather than selecting these values manually, a common approach is to formulate an additional optimization procedure. Herein, the Tree-structured Parzen Estimator (TPE) algorithm [3] is employed, implemented in the Optuna library [105], and target the minimization of the overall loss function computed for the test dataset. In other words, reasonable ranges for the hyperparameters are considered and then the training phase for each step of the optimization process (in total 100 iterations have been considered) is repeated. The ranges considered are: batch size ($N_b \in [3, 32]$), learning rate ($\in [10^{-5}, 10^{-1}]$) and regularization coefficient ($\beta \in [10^{-10}, 10^{0}]$). Figure 4.3.1 shows the iterations performed by the optimizer as function of the objective function (the loss function computed for only the validation dataset). The final hyperparameters selected are: batch size $N_b = 3$, learning rate = 1.56×10^{-3} and $\beta = 4.84 \times 10^{-10}$.

There is one last, critical hyperparameter in the AE architecture, the dimension of the latent space N_{ls}. While this can be treated easily as part of the optimization described above, a different strategy is here used. In section 3.4 it was established a link between latent representation and parametric characteristics of the database; here the same strategy is adopted, keeping the dimension of the latent space as small as possible for interpretability and to facilitate the use of the AE as a generative model. Various tests were conducted with N_{ls} values of up to 10, but the outcomes did not exhibit significant alterations. Consequently, a choice of $N_{ls} = 3$ was made for all subsequent investigations.

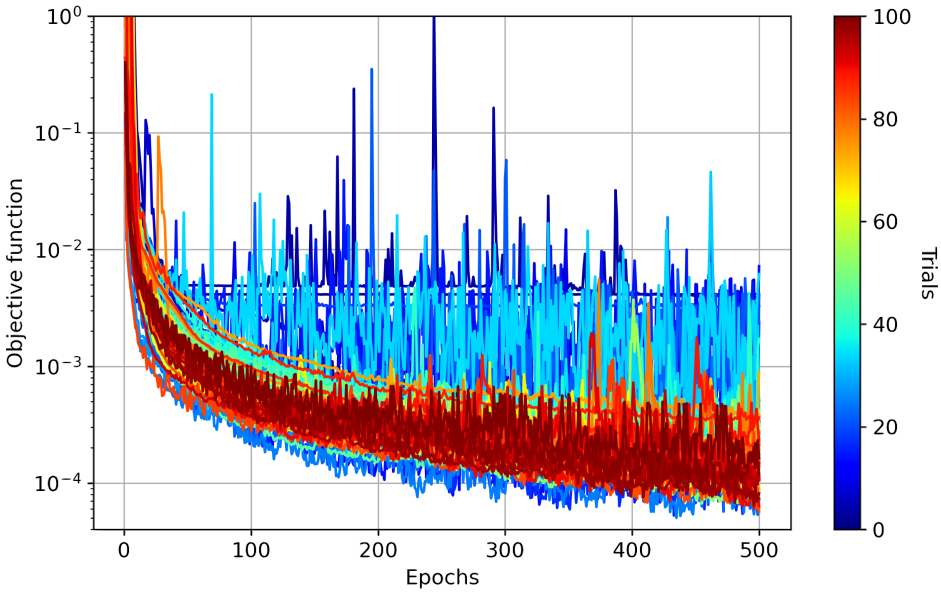


Figure 4.3.1: Hyperparameter tuning: objective value as function of the epochs for the 100 iterations (trials) performed by the TPE algorithm [3].

For completeness is worth noting that the details of the architecture described in the previous section, such as the number of convolutional layer for example, could also be managed via hyperparameter tuning. Here instead, the architecture is treated as corresponding to the *form of the model* and explicitly study the (internal) uncertainty associated to these choice. This is because in many cases architecture choices are dictated by experience, interpretability or other constraints or considerations. In any case, the approach presented in the following section provides a flexible framework to manage the hyperparameters as either uncertainties or tuning parameters.

4.4 Deterministic vs. stochastic autoencoders

The AE approach described above is *deterministic* in the sense that, after completion of the learning phase, given an unseen input x the encoder generates repeatable values corresponding to the latent vector V. This low-dimensional representation $\mathbb{R}^n \rightarrow \mathbb{R}^{N_{ls}}$ corresponds to a data compression ($n >> N_{ls}$). On the other hand, for every realization of the latent vector the decoder *generates* a vector $\hat{x}$. These two modes of operations will be analyzed separately; the first corresponds to representation learning or feature extraction approach, will refer to it simply as AE. The second corresponds to a prediction task, referred to as G-AE (generative AE). More details regarding the generative AE are reported in the following section.

As mentioned earlier, a stochastic variant of AEs is also used often in the literature espe-

cially for generative tasks: the VAE. In this case the latent vector consists of random variables (typically Gaussian i.i.d. variables); after the learning phase is completed, realizations from the stochastic latent space enable the construction of a probabilistic representation of $\hat{x}$. VAE have been introduced to reduce the effect of noise in the training data, and, therefore, are not directly targeting the quantification of the uncertainty in the predictions induced by the choice of the dataset or the network architecture. Furthermore, the normality assumption in the representation of the latent variable is not justified in general, and introduces further assumptions that cannot easily be verified. Clarity on this matter will be achieved later when the presence of stochasticity in the latent space is illustrated in relation to internal uncertainty.

4.5 Uncertainty Quantification

ML approaches produce predictions that are affected by both assumptions (choices) used in the definition of the computational architecture and by the data used during the learning phase (internal uncertianty). The AE hyperparameters are set as described above 4.3. The AE is used to predict the aerodynamic performance of NACA 0012 airfoil (i.e. $t/c = 12\%$) at $M_\infty = 0.15$, $Re_\infty = 6.0 \times 10^6$ in a range of angles of attack $\alpha \in [-30°, 50°]$. Although the AE generates the entire flow field around the airfoil in terms of pressure and velocity components, the analysis is centered on the predictions of the lift coefficient $C_l(\alpha)$ as the quantity of interest. This is however computer as the integral of the pressure distribution on the airfoil surface, and therefore requires an accurate representation of the predicted field to be successful.

Assessing the corresponding sensitivity of the computed results is not straightforward, although in the literature multiple strategies have been introduced. First it is important to specify how the uncertainties enter the modeling process.

4.5.1 Sources of uncertainty

ML algorithms rely on data for training, but also on many user-defined algorithmic and architectural choices, even when hyperparameter tuning is used to optimize their performance. In this section, a comparison is made regarding the impact of three distinct types of uncertainty on the predictions generated by the previously introduced AE. The effect of variability induced by initialization and training batches, the impact of changes in the architecture and, finally the prediction changes induced by the size of the dataset are explored. In all instances, an ensemble of trained AEs is considered. The evaluation includes an assessment of the error associated with the mean prediction concerning the ground truth, as well as the extraction of a confidence interval from the ensemble using a bootstrap method [106].

Aleatory uncertainties

The first investigation is focused on the learning process, and specifically the effect of the selection of the batches. Batch learning has emerged as a common technique to accelerate the convergence of the optimization process; however, especially for small datasets, it is possible that the diversity within a batch remains limited thus leading to potential overfitting. A further source of randomness in the learning process is the initialization of the network weight, which is based on a random seeding. Together the weight initialization and the choice of batches correspond to aleatory sources, i.e. sources of randomness. To characterize their effect an ensemble of trained AEs consisting of 10, 50 and 200 members was considered. Both the errors with respect to the ground truth and the 95% confidence interval in the prediction computed using the bootstrap method are reported.

Figure 4.5.1 is a summary of the results obtained. The main plot (bottom) shows the lift curve for the NACA 0012 reported in terms of the ensemble mean and the confidence intervals; it is compared to the ground truth which in this case corresponds to the RANS predictions. The upper plot reports only the size of the confidence interval to illustrate more quantitatively the effect of the stochasticity in the learning phase. In Figure 4.5.1 the symbols on the top horizontal axis represent the datasets available for the learning phase (cfr. Figure 4.2.2) with the closest geometry, i.e.the NACA 0008 and NACA 0016 airfoils. The results illustrate that the (mean) AE predictions are in good agreement with the ground truth in the overall range. As expected, the uncertainty in the extrapolatory region ($\alpha > 30°$) increases as the angle of attack is increased. Interestingly, the confidence interval in the interpolatory region $\alpha \in [-15°, 0°]$ also increases towards the center of the data gap described earlier, and exceeds the corresponding variability in the extrapolatory region. This observation is consistent with the conclusion that distance of the prediction scenario from the learning cases is the factor controlling the confidence in the predictions. Specifically, the increase in standard deviation is far more abrupt in the interpolatory region compared to angles of attack $\alpha > 30°$. This might be due to the different aerodynamical characteristics of the two region; in the extrapolatory region the flow is already highly separated and differences in the angle of attack are not producing considerable changes in the behavior of the solution. On the other hand in the interpolatory region, the flow is transitioning from a linear regime (at very low angle of attack) towards the stall region, and flow separation starts to appear on the airfoil surface. To confirm these observation and provide more quantitative information about the AE predictions, are here reported the pressure coefficient on the airfoil surface at selected angles of attack.Specifically, $\alpha = -8°$ is located at the center of the data-gap; $\alpha = 17°$ is in the stall region, close to the maximum lift coefficient; and $\alpha = 40°$ is a deep-stall regime, where the flow is massively separated from the airfoil surface.

In agreement with the previous analysis, the $\alpha = -17°$ shows the smallest uncertainty. An important observation from the results in Figure 4.5.2 is the strong correlation between high standard deviation and high discrepancy between the AE prediction and the ground truth. For example at $\alpha = -8°$ the largest discrepancy is observed on the upper surface of the airfoil near

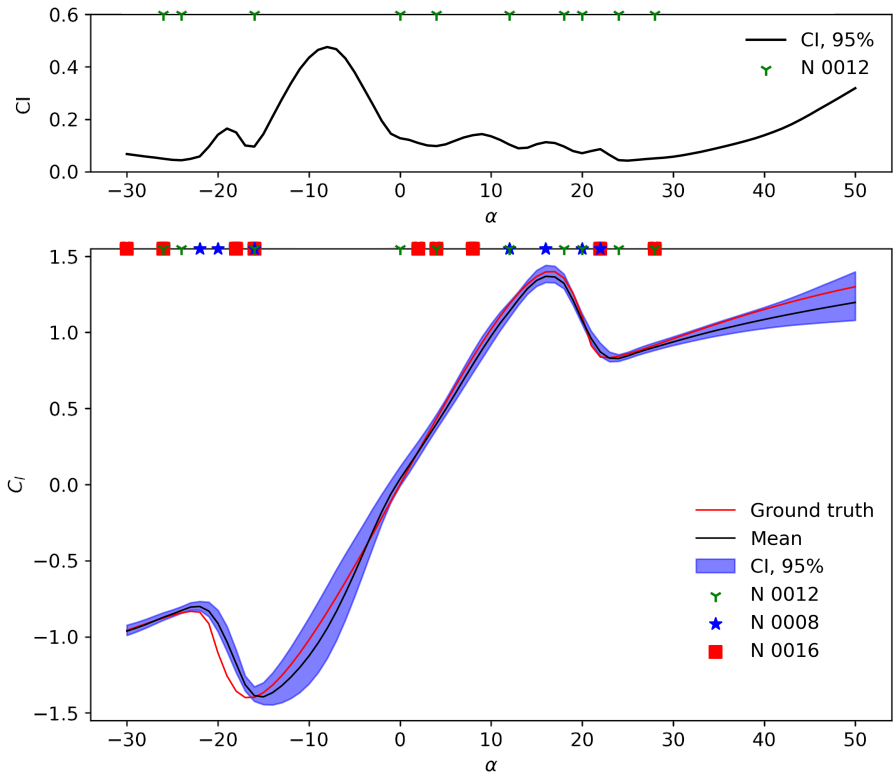
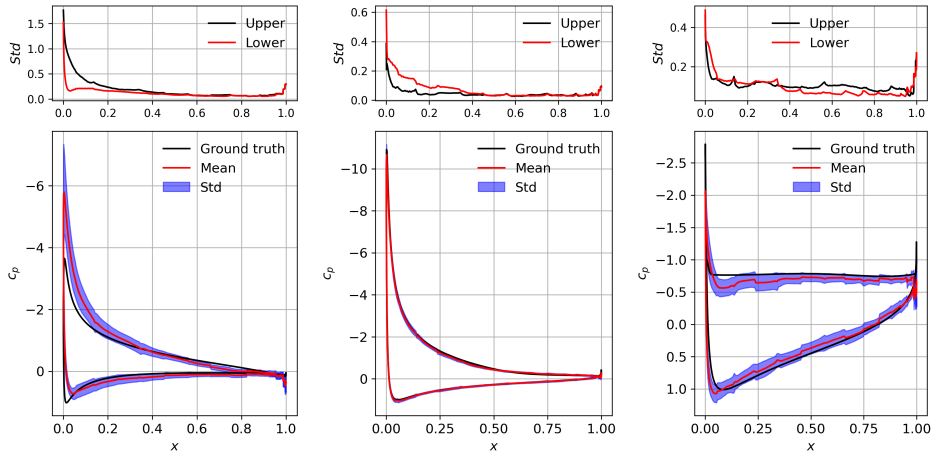


Figure 4.5.1: Predictions obtained using the trained decoder: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$. 95% confidence interval. (top) and mean (bottom) of the lift curve as function of α. The mean prediction and the confidence interval correspond to an ensemble of 50 AEs. The symbols on top of the figure represent the closest data in the training set.



(a) $\alpha = -8°$ **(b)** $\alpha = 17°$ **(c)** $\alpha = 40°$

Figure 4.5.2: Predictions obtained using the trained decoder: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$. Standard deviation (top) and mean (bottom) of pressure coefficient on the airfoil surface. The mean prediction and the standard deviation correspond to an ensemble of 50 AEs trained with different initial weights and batches.

the leading edge, while for $\alpha = 17°$ the largest discrepancy is on the lower surface. To confirm the correlation between the ensemble standard deviation and the actual prediction error, a comparison of the decoder-generated flow field around the NACA 0012 at $\alpha = 40°$ is reported; specifically the Mach number contours can be visualized in Figure 4.5.3, in terms of the local percentage error obtained comparing the AE predictions and the ground truth and standard deviation of the ensemble predictions. Again the correlation between the two quantities is promising because it enables to establish that the ensemble standard deviation (which does not require knowledge of the ground truth) is a reasonable proxy for the actual error. Figure 4.5.4 and Table 4.5.1 report quantitative results comparing ensembles of different sizes (from 10 to 200 members). The results illustrate that the increased size of the ensemble does not lead to a reduction

ensemble members	$\alpha = -8^o$ Error (95% CI)	$\alpha = 17^o$ Error (95% CI)	$\alpha = 40^o$ Error (95% CI)
10	0.111 (0.564)	0.051 (0.142)	0.022 (0.117)
50	0.107 (0.477)	0.035 (0.110)	0.066 (0.139)
200	0.046 (0.447)	0.045 (0.139)	0.080 (0.156)

Table 4.5.1: Effect of Aleatory Uncertainty in the predictions obtained by ensemble of AEs at different angles of attack.

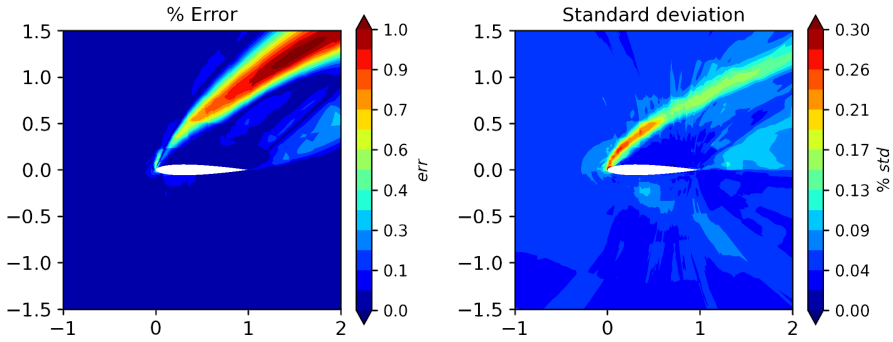


Figure 4.5.3: Predictions obtained using the trained decoder: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$, $\alpha = 40°$. Percentage discrepancy between AE prediction and the ground truth (left) and standard deviation of AE ensemble (right).

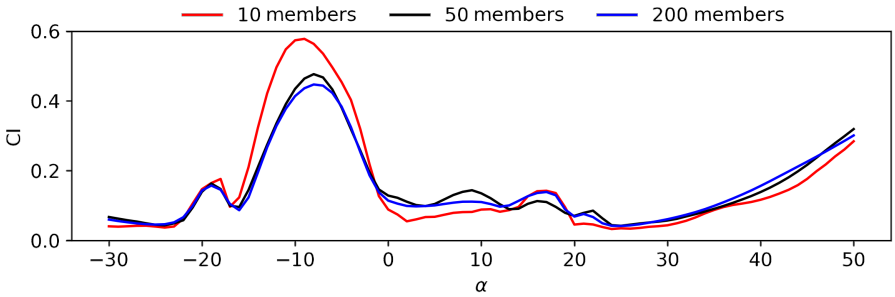
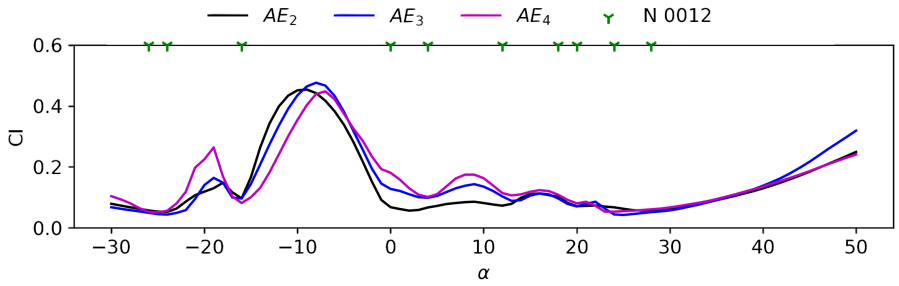


Figure 4.5.4: Amplitude of (95%) confidence interval of the lift curve as function of α for the NACA 0012 airfoil, $Re_\infty = 6.0 \times 10^6$ and $M_\infty = 0.15$. Comparison between 10, 50 and 200 members.

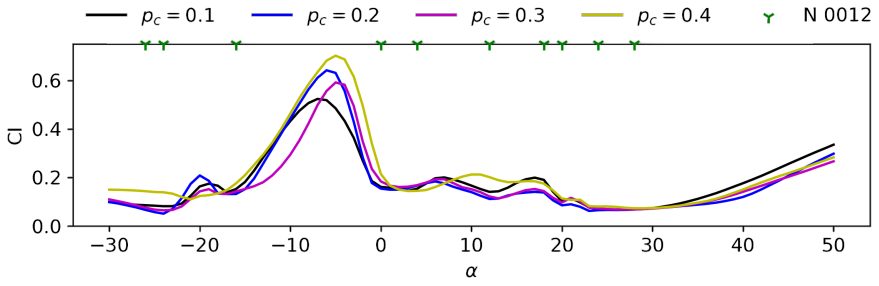
of the confidence intervals; this is consistent with the expectation that both in the interpolatory and extrapolatory region, the lack of training data is the key reason for the discrepancy.

Epistemic Uncertainty

Deep networks and AEs use a network consisting on multiple layers with a very large number of parameters to be estimated during the learning phase. For a given dataset, the choice of the optimization process, the loss function and the presence of weight regularization lead to potentially different trained networks. In addition, when convolutional layers are adopted, the choice of the kernels and the number of layers also introduces potential sensitivities.



(a) Effect of number of convolutional layers



(b) Effect of dropouts

Figure 4.5.5: Predictions obtained using the trained decoder: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$. Ensemble 95% confidence intervals obtained using 2, 3, and 4 convolutional layers (top) and different percentage of dropout connections (bottom).

In this section the sensitivity of the AE predictions to the number of convolutional layers are first considered. The basic architecture, introduced earlier, uses three layers, the effect of using two and four layers as well is investigated. The expectation is that less layers lead to a broader kernel that increases the overall smoothness of the features identified by the AEs; viceversa more layers attempt to represent more local features. The overall consequence is that the prediction obtained using more layers tend to be less accurate in the regions with sufficient data (say in the range of $a \in [0 : 12]$) while not providing any real advantage for interpolatory and extrapolatory tasks.

size of dataset	$\alpha = -8^\circ$ Error (95% CI)	$\alpha = 17^\circ$ Error (95% CI)	$\alpha = 40^\circ$ Error (95% CI)
30	0.053 (0.396)	0.181 (0.104)	0.077 (0.152)
44	0.107 (0.477)	0.035 (0.110)	0.066 (0.139)
59	0.098 (0.501)	0.023 (0.080)	0.057 (0.161)

Table 4.5.2: Effect of the size of the dataset in the predictions obtained AEs at different angles of attack.

The second sensitivity study is related to the effect of regularization in the learning phase. A popular technique in this context is based on *dropouts*, in which a random set of network connections are cut off during the training phase. This is effectively a regularization that enforces sparsity in the optimization process and limits overfitting [107]. In this case the user specifies the percentage of the total connections that are severed (selected randomly) and the learning phase proceeds as usual. In the original AE no dropouts was employed, but the results indicate that the overall sensitivity is small if the most aggressive case in which 40% of the connections are *dropped* is excluded. There is again a large uncertainty (confidence interval) in both the region of extrapolation and interpolation.

4.5.2 Dataset size sensitivity

In the context of this investigation, the focus is directed toward the training of ML models using comparatively small datasets, which are more representative of the data limitations encountered in real-world engineering applications. The results presented thus far are based on a dataset consisting of 44 cases. To assess the results' sensitivity to dataset size, two additional datasets, one comprising 33 solutions and the other containing 59 solutions, are considered. As before, the distribution of training and validation cases is split as 85% and 15%, and the coverage of the parameter space, and specifically the data gap designed in the range $\alpha \in [-15^\circ, 0^\circ[$ remains unchanged.

The results presented in Figure 4.5.6 show the expected improvement of the predictions with a standard deviation decreased as compared to Figure 4.5.1. However, it is quite striking how the AE prediction quality in the extrapolatory and the interpolatory regions has not improved! This results confirms that the discrepancy index reported before (see Figure 4.2.1) as a measure of the sufficient size of the dataset is essentially correct as it focuses on the value added by each additional case; however, this measure does not provide information about the potentially limited coverage of the parameter space. The error of the prediction decreases with the number of samples in the database (Table 4.5.2), especially in the region close to the maximum lift ($\alpha = 17^\circ$) as expected. In the interpolatory and extrapolatory regions the error reduction is not accompanied with an increase in confidence (reduction of the confidence interval) pointing to that fact that additional data does not provide more information. This is even more evident from Figure

4.5.6 were the largest dataset ($N = 59$) leads to higher confidence in the range $0° \leq \alpha \leq 20°$ but lower confidence both in the interpolatory and extrapolatory regions.

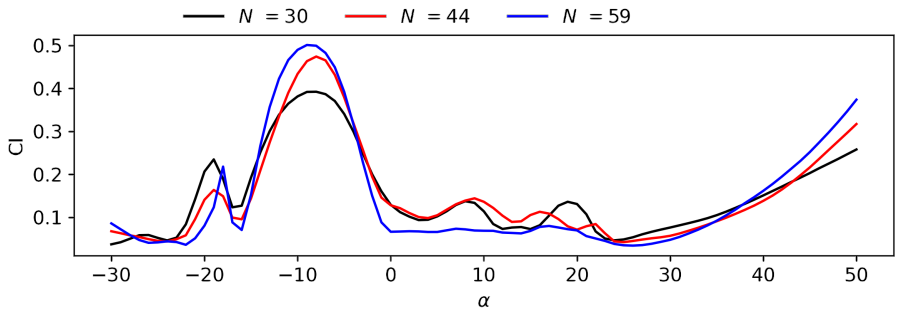


Figure 4.5.6: Amplitude of confidence intervals of the lift curve as function of α. Comparison between different dataset size.

4.5.3 Uncertainty in the latent variables

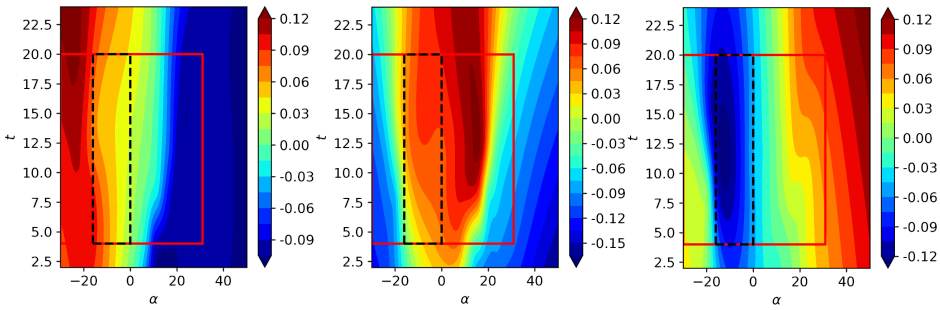
The results presented in the previous section illustrate the effect of the internal uncertainty on the AE predictions. It is also insightful to assess the variability induced by the identified uncertainty sources on the latent variables. This is effectively also a demonstration of how the present approach differs from VAEs in which an ansatz on the latent space representation (e.g. gaussian variables) is used.

To quantify the uncertainty in the latent space, it is necessary to standardize the latent variables across the AEs in the ensemble. This necessity arises from the observation that, during the training of AEs multiple times, the latent space, while maintaining its fundamental compression properties, can undergo rotations, translations, and scaling. These variations may be attributed to diverse factors, such as the random initialization of network weights or architectural modifications.

However, if each training instance consistently converges to a global minimum, the overall shape and relative positioning of the latent variables remains consistent across all ensemble members. To quantify the mean and standard deviation of the latent variables, the transformation referred to as *Procrustes* analysis [108, 109] is employed.

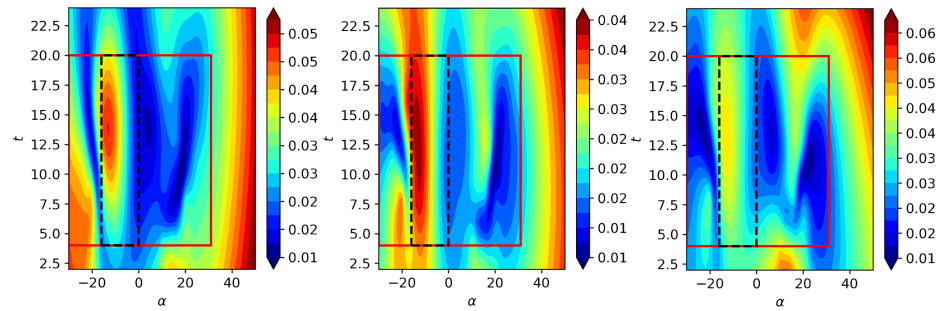
The Procrustes analysis seeks to achieve an optimal overlap between two datasets through a combination of rotations, translations, and uniform scaling. The objective is to find a rotation matrix, a translation vector and a scaling factor, in order to minimize the sum of the squared differences between corresponding data points in the two input datasets. Figure 4.5.7 displays the mean values of the variable mappings. The training dataset's boundaries are highlighted by the red rectangle, while the data-gap region is delineated by the black dashed lines.

To examine the variability in the latent space shape, the standard deviations of the latent variables are presented in Figure 4.5.8. Notably, the highest degrees of variability are observed



(a) V_1 **(b)** V_2 **(c)** V_3

Figure 4.5.7: Latent space uncertainty: means of the latent variables. The red rectangle marks the boundaries of the dataset region; black dashed rectangle marks the boundaries of the data-gap region.



(a) V_1 **(b)** V_2 **(c)** V_3

Figure 4.5.8: Latent space uncertainty: standard deviations of the latent variables. The red rectangle marks the boundaries of the dataset region; black dashed rectangle marks the boundaries of the data-gap region.

within the data-gap and the extrapolation region, while inside the red rectangle, the standard deviation assumes the lowest values.

4.6 Gaussian Process Regression

At this stage it is useful to contrast the results obtained in the previous section with a methodology that is popular in practical applications, Gaussian Process regression [110]. The approach used here is to *feed* the GPR with the same dataset that is used earlier and aim at the same prediction task: given the flow fields corresponding to the database in Figure 4.2.2 extract the pressure distribution on the surface of the airfoil and compute the lift coefficient as a function of the angle of attack for an unseen airfoil geometry.

Here only a subset of the results is provided. The details of the GPR used and sensitivity to the choices are reported in the D.

The lift curve is reported in Figure 4.6.1 and overall shows a reasonable agreement with the ground truth. However, the GPR appear to provide very high-confidence in the results even in the region where data is missing and only increases deep in the extrapolatory region. The overall error remains small and is not well correlated with the size of the confidence interval; for example the GPR fails to predict the maximum lift and the corresponding stall but the confidence bands are not changing. The GPR predictions are analyzed in more details by looking at the pressure distributions on the surface. This is reported in Figure 4.6.2 (that can be compared directly to Figure 4.5.2). The results clearly illustrate that the predictions are not very accurate and the standard deviation remains largely constant, instead of providing a proxy for the prediction error. A final result is showed in Figure 4.6.3 were the error in the MAch number flowfield around the airfoil and the GPR standard deviation are reported (compared to Figure 4.5.3).

4.7 Predictions with quantified uncertainties

, the predictive capabilities of the trained AEs have been established and illustrated the strategy to assess the uncertainties present in the model in a controlled setting were the ground truth is precisely known. In a realistic application, other uncertainties can be active, for example due to limited knowledge of the operating scenarios or in the presence manufacturing tolerances (external uncertainties). In this section, the focus is directed to the AE predictions of the aerodynamic characteristics of a NACA airfoil (NACA 0010) in the presence of geometrical uncertainty, represented as a thickness variability of $\pm 5\%$. The results are compared to the *true* uncertainty predicted by carrying out simulations using the RANS solver.

The objective is to compare the relative importance of the uncertainty in the physical input (thickness) with both the aleatory and the epistemic sources reported before (internal uncertainties due to AE architecture, the hyperparameters and the training data). It is known that changes in airfoil thickness will only affect the stall regions and therefore, this application is in-

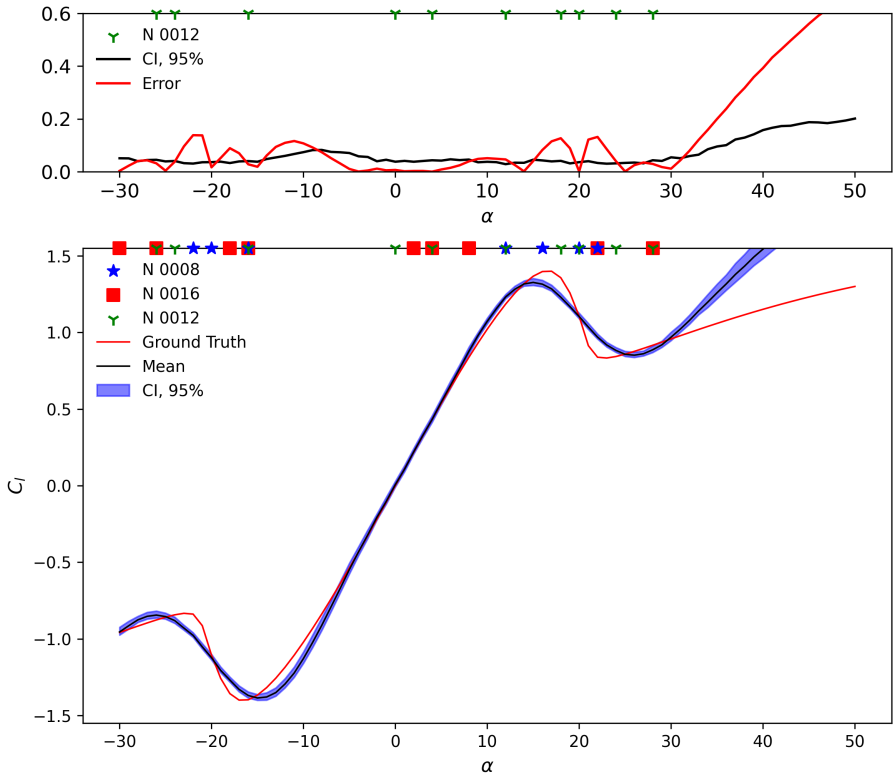
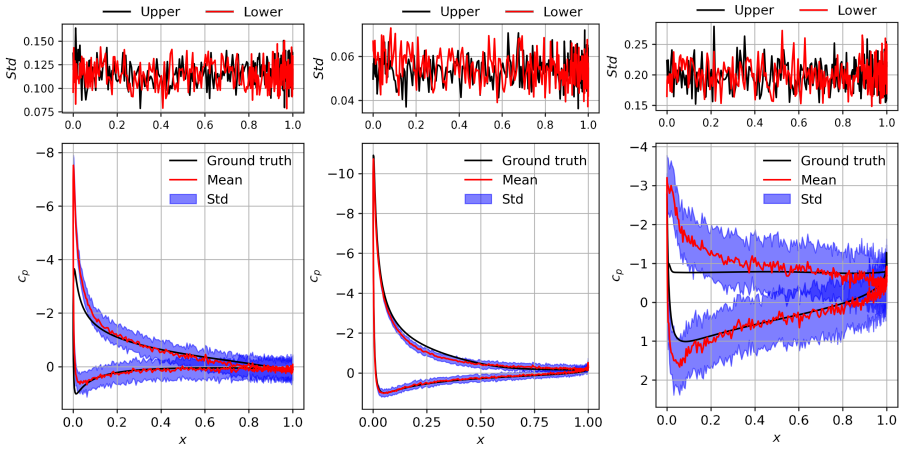


Figure 4.6.1: Predictions obtained using a GPR: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$.



(a) $\alpha = -8°$ **(b)** $\alpha = 17°$ **(c)** $\alpha = 40°$

Figure 4.6.2: Predictions obtained using GPR: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$. Standard deviation (top) and mean (bottom) of pressure coefficient on the airfoil surface. The mean prediction and the standard deviation correspond to an ensemble of 50 samples.

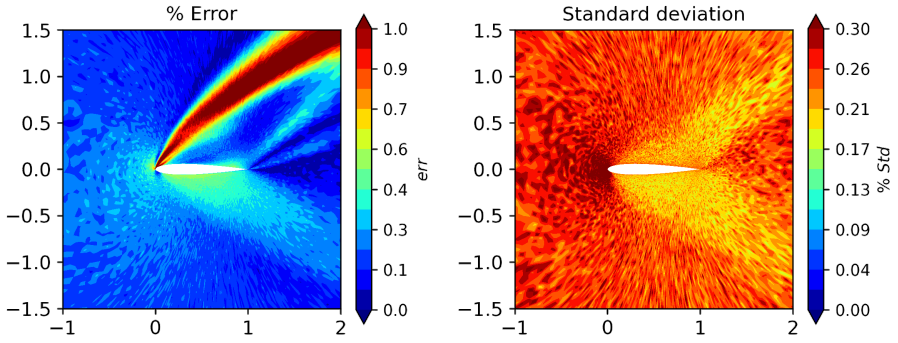


Figure 4.6.3: Predictions obtained using a GPR: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$, $\alpha = 40°$. Percentage discrepancy between GPR prediction and the ground truth (left) and standard deviation of GPR ensemble (right).

	$\alpha = -8^\circ$ Mean (95% CI)	$\alpha = 17^\circ$ Mean (95% CI)	$\alpha = 40^\circ$ Mean (95% CI)
AE	-1.0228 (0.0218)	1.266 (0.0508)	1.102 (0.00517)
GT (RANS)	-0.834 (0.00311)	1.307 (0.0437)	1.175 (0.00703)

Table 4.7.1: Uncertainty in the geometry of the NACA 0010 (represented as thickness variability of $\pm 5\%$). Lift coefficient comparison between the ground truth (GT) using an ensemble of RANS simulations and the AE prediction.

tended as a demonstration that the AE can also faithfully represent the variability in the physical inputs. For this reason, an ensemble of 50 RANS computations (for each angle of attack) is also carried out to quantify the ground truth uncertainty. It is worth noting that to build the entire lift curve (i.e. for $\alpha \in [-30^o : 50^o]$) a very large number of RANS computations would be required, consequently, the focus was primarily directed to the comparisons of the three representative angles of attack, as previously detailed: $\alpha = -8^\circ, 17^\circ, 40^\circ$. On the other hand AE predictions with quantified uncertainty are very inexpensive and therefore confidence intervals can be quickly built for the entire range of angles of attack under consideration. A further point is that the ensemble of RANS computations captures the variability in the input, but not the internal uncertainty that is induced by the modeling choice employed (i.e. the RANS turbulence model). An investigation of this source of uncertainty is outside the scope of this work, but it is worth noting that the approach developed in [111] consistently shows very low confidence in the predictions for angles of attach approaching stall (cfr. Figure 15 [111]). The main results of the comparisons between uncertainties obtained using the AE and the RANS computations are reported in Table 4.7.1. It is reassuring that in the stall region ($\alpha = 17^o$) the predictions are in agreement both in terms of the mean lift coefficient and the corresponding confidence interval. This is the region were the data-driven model is sufficiently trained as indicated by the relatively small internal uncertainty. In the extrapolation region ($\alpha = 40^\circ$) the AE is in good agreement with the ground truth and correctly indicates that the sensitivity to the variability in the physical input is not very important; this is consistent with the ground truth. The AE prediction however, are endowed with a measure of the internal uncertainty and, in this case, the corresponding confidence interval overwhelms the variability induced by the physical input, indicating that the trust in that prediction is somewhat limited. By far the most interesting situation corresponds to the pre diction of the $\alpha = -8^\circ$ case. It is expected that the thickness uncertainty has very little influence on the predictions, and this is certainly confirmed by the RANS computations. As sown earlier, the internal uncertainty of the AE prediction is very high in this region; the results illustrate that the AE correctly distinguishes between the two sources of uncertainty, with the confidence interval corresponding to the variability in the geometry being an order of magnitude smaller than the internal uncertainty. Overall it is clear that in the entire range of angles of attack (as reported in Figure 4.7.1) the only region were the external uncertainty is high is close

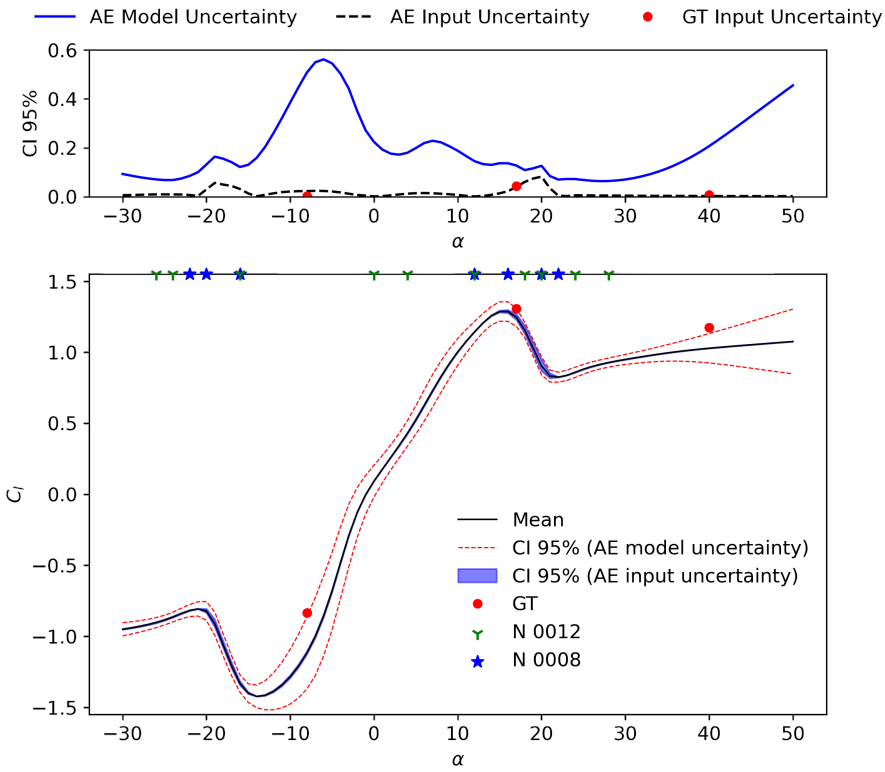


Figure 4.7.1: Predictions obtained using the trained decoder: NACA 0010, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$. 95% confidence interval (top) and mean (bottom) of the lift curve as function of α. Comparison between GT and AE.

to the stall. The AE properly distinguishes between external and internal uncertainty and even in the absence of ground truth data, it provides useful insights into necessary improvements of the training dataset.

Conclusions

An innovative technology has been explored in the field of CFD: Machine Learning. This dissertation conducted an exploratory analysis on the potential of ML for aerodynamic analyses. While it is rapidly becoming a common practice that complements more classical methods, there are still some open questions about this technology.

In a first part of the present study, the Gaussian Mixture, a widely recognized clustering algorithm, was employed in a novel context: the physical-based domain decomposition. Previously, this process relied on deterministic sensors, which posed certain difficulties and necessitated user expertise. The innovation introduced in the present research encompasses the following key aspects:

- a ML algorithm was successfully developed to automate the domain decomposition process without requiring any manual intervention. This algorithm was applied to the aerodynamic drag breakdown by far field methods for airfoils, wings and aircraft in subsonic and transonic operational regimes. Comparative analysis against classical methodologies revealed the significant potential of this algorithm, effectively overcoming many of the limitations associated with the existing approaches.
- Furthermore, it is worth noting that this algorithm is not confined to the aerodynamic force decomposition. Its adaptability allows for seamless integration to a wide array of fields. For instance it can be easily embedded into iterative optimization loops, adaptive mesh refinement and more.

In a second part, a deep-learning algorithm is investigated: convolutional autoencoder. Beyond assessing the capability of AEs in accurately predicting the aerodynamic field around wing sections, this dissertation explored several key questions in pursuit of answers:

- *Is it possible to provide an interpretation of the learning procedure of ML algorithms, particularly the latent space of AEs?*

  Indeed, interpreting the latent space of AEs and other ML models is an ongoing challenge, and the specific techniques and level of interpretability required can vary depending on the application. Balancing model complexity and performance with interpretability is crucial to ensure that ML models can be effectively used in practical real-world scenarios while providing insights into their decision-making processes.

The results presented in this work show that a form of *physical imprinting* is observed in the latent space of AEs. Specifically, in case of airfoil aerodynamics, AEs have the capability to extract and encode physical information from the input flow field, with a strong correlation to aerodynamic forces and moments. Notably, this characteristic remains consistent across various architectural configurations, dataset dimensions, and complexity of the physical phenomenon, underscoring the potential of AEs for capturing relevant physical features in the input data.

As a result, while this was showed in the context of an aerodynamic phenomenon, where numerous theories have been developed over the years, AEs can serve as a powerful tool to uncover the underlying physics of phenomena that are not yet well understood.

- Another intriguing challenge within the deep-learning framework is the substantial data requirements inherent to such algorithms. The dissertation showed the possibility of training ML models with a reasonable level of accuracy using a limited dataset. An iterative algorithm for an *adaptive database generation* was developed for this purpose. The algorithm was successfully applied to train an AE to predict the flow field around NACA airfoils at different Reynolds number. This is a crucial step in extending these models to high-fidelity aerodynamics databases (LES and DNS), where the computational cost for the dataset construction can be a significant obstacle.

- Open challenges in uncertainty quantification of ML models are investigated. In particular, quantifying the uncertainty of AEs using the ensemble method. A specific focus of investigation involves the effect of intrinsic sources, and specifically the modeling choices behind the architecture and the hyperparameter selection. Additionally, an analysis is conducted to gauge the influence of data availability. It was concluded that the AEs correctly identify regions were the prediction confidence is low. Furthermore, a noteworthy correlation emerged between the standard deviation in the AE predictions and the extent of error in comparison to the ground truth. This correlation confirms that even in the absence of the latter, the AEs offer valuable insights into the limitations inherent to the model itself. Finally, the study of the effect of physical input uncertainty, showed the ability of the AE ensemble in discerning between the different sources of uncertainty.

  While the current findings pertain to aerodynamic prediction tasks and rely on a training datasets based on RANS predictions, the fundamental concept of systematically characterizing diverse sources of uncertainty and conducting comparative analyses of their impact possesses a broader applicability. This approach can be effectively extended to diverse domains and challenges beyond aerodynamics, offering a versatile framework for understanding and addressing uncertainty in various contexts.

Overall, this work aims to shed light on the opportunities and challenges presented by ML in the field of aerodynamics and Fluid Mechanics.

Most of the research findings presented in this dissertation were published in academic journals and presented at international conferences. Further details regarding these publications are reported in the *Published Excerpts of Present Dissertation* located at the end of the thesis.

Appendices

A

Mathematical details of GMM

Often it is not possible to model the data using a simple distribution. Therefore, the data can be modeled as a *mixture* of many components of simple parametric form, for instance Gaussian. An example of a Gaussian mixture is reported in figure A.0.1.

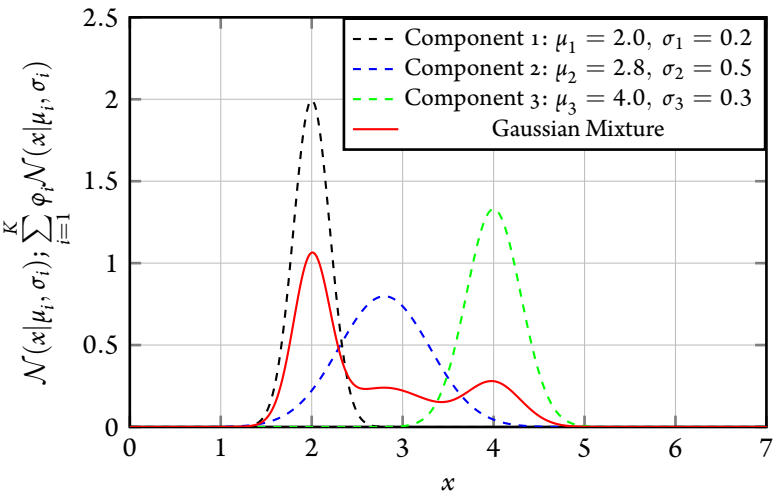


Figure A.0.1: Example of an Univariate Gaussian Mixture distribution built up using 3 components, with $\varphi_1 = 0.5$, $\varphi_2 = 0.3$ and $\varphi_3 = 0.2$.

The red curve represent the mixture of the Gaussian distributions composed by a combination of 3 components. In this case we are saying that we want to find 3 clusters, so the parameters describing the three components need to be found. For a one-dimensional model (Univariate Gaussian Mixture) the distribution is:

$$p(x) = \sum_{i=1}^{K} \varphi_i \mathcal{N}(x|\mu_i, \sigma_i) \tag{A.1}$$

with

$$\mathcal{N}(x|\mu_i, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right) \tag{A.2}$$

where K is the number of components, μ_i and σ_i are respectively the mean and the variance of the i^{th} component and φ_i is the weight (mixing proportion) of the i^{th} component constrained by $\sum_{i=1}^{K} \varphi_i = 1$.

In case of a multi-dimensional model (Multivariate Gaussian Mixture Model), the distribution is:

$$p(x) = \sum_{i=1}^{K} \varphi_i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{A.3}$$

with

$$\mathcal{N}(x|\mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K|\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right) \tag{A.4}$$

where Σ is the covariance matrix.

φ_k, μ_k and Σ_k need to be found considering the x_i data assigned to the C_k cluster. The parameters are learned using the *Expectation-Maximization* (EM) iterative algorithm, which is a general method of finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given data set [63]. The algorithm is divided in two steps: the expectation (E) and the maximization (M). The E step computes the expectation of the assignments of each data point x_i to the C_k cluster using the model parameters φ_k, μ_k and Σ_k. The M step maximize the expectation computed in the E step with respect to the model parameters. At the end of the M step, the parameters φ_k, μ_k and Σ_k are updated and given in input to the E step again. The algorithm iterates until the convergence is reached. Before the E and M steps, there is a random initialization of the model parameters. For clarity lets consider an Univariate Gaussian Mixture Model, where φ_k, μ_k and σ_k are the model parameters of the Gaussian distribution.

- **E step**: computes the probability

$$p(C_k|x_i, \hat{\varphi}, \hat{\mu}, \hat{\sigma}) = \frac{\hat{\sigma}_k \mathcal{N}(x_i|\hat{\mu}_k, \hat{\sigma}_k)}{\sum_{j=1}^{K} \hat{\varphi}_j \mathcal{N}(x_i|\hat{\mu}_j, \hat{\sigma}_j)} \tag{A.5}$$

  for a more compact notation $p(C_k|x_i, \hat{\varphi}, \hat{\mu}, \hat{\sigma}) = \hat{p}_{ik}$ is used in what follows.

- **M step**: updates the model parameters

$$\hat{\sigma}_k = \sum_{i=1}^{N} \frac{\hat{p}_{ik}}{N} \; ; \quad \hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{p}_{ik} x_i}{\sum_{i=1}^{N} \hat{p}_{ik}} \; ; \quad \hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \hat{p}_{ik} (x_i - \hat{\mu}_k)^2}{\sum_{i=1}^{N} \hat{p}_{ik}} \tag{A.6}$$

The two steps are performed until convergence, and the model parameters are computed.

In order to compute the probability that each data point belongs to a particular cluster C_i, the Bayes' theorem is used:

$$p(C_i|x) = \frac{\varphi_i \mathcal{N}(x|\mu_i, \sigma_i)}{\sum_{j=1}^{K} \varphi_j \mathcal{N}(x|\mu_j, \sigma_j)} \tag{A.7}$$

In case of a Multivariate Gaussian Mixture Model the procedure is analogous.

Appendix A. Mathematical details of GMM

B

Aerodynamic Drag Breakdown

The computation of the aerodynamic force by stress integration on the body surface (*near field* method) allows for the decomposition in friction and pressure force, but does not give any information on the splitting among viscous, lift-induced and wave contributions which, on the contrary, is obtainable by *far field* methods, so called because based on formulae derived from the integral momentum equation.

Drag-breakdown researches, so far, can be grouped into two families: thermodynamic and vorticity-based force breakdown. The former is based on Oswatitsch's entropy drag concept, and usually decomposes the aerodynamic force into reversible and irreversible parts by linearized process splitting, then by momentum equation to realize the drag breakdown [68, 112]. The irreversible drag is identified by selecting the component associated to entropy production. Thermodynamic methods are widely adopted in industrial and research environment and already contributed to the design of last generation of transonic transport aircraft. They are limited to the computation and analysis of the force components of irreversible nature, therefore cannot compute and analyze lift and the lift-induced drag can only be derived in an indirect way by subtracting the irreversible drag to the near field force.

Vorticity-based methods start from a general formulation of the vortex-force theory. Wu et al. [113] evidenced the role of the Lamb vector as the local flow structure responsible for the aerodynamic force generation. Mele and Tognaccini [114] simplified the theory in the compressible and turbulent case, highlighting the generation of the whole aerodynamic force in

the vortical part of the flow only even in compressible flows. The potential of these analyses is evidenced by the recently developed unified force theory proposing a single force expression valid in steady viscous subsonic, transonic and supersonic flow (Jukowskij-Filon theorem) [115, 116].

Recently, Schmitz in [117] proposed an alternative method of drag decomposition, using partial-pressure fields, and demonstrating that it is equivalent to classical far field analysis.

Here the fundamental formulae of the thermodynamic method adopted are just recalled.

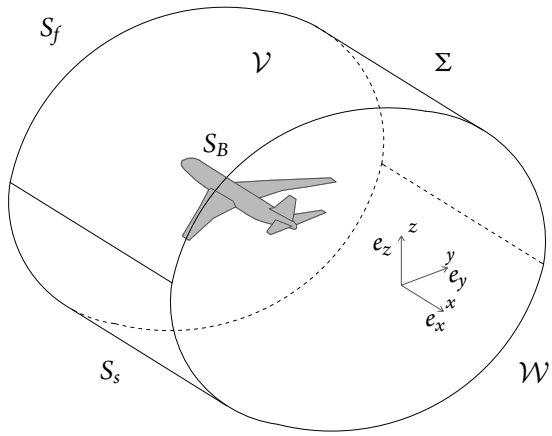


Figure B.0.1: Sketch of fluid domain considered for the aerodynamic force analysis.

A turbulent, steady high-Reynolds number compressible flow around an aircraft configuration is considered. Assuming a cartesian reference system with the x axis aligned with the asymptotic velocity, a straightforward application of the momentum balance equation provides the far field drag expression (see figure B.0.1):

$$D_{far} = -\int_{\Sigma} [\rho u(V \cdot n) + (p - p_\infty)n_x] \, \mathrm{dS} \,, \tag{B.1}$$

where ρ is the density, $V = [u, v, w]^T$ is the local velocity, p is the pressure and subscript ∞ specifies free-stream conditions. Σ is the outer boundary of the computational domain, $n = [n_x, n_y, n_z]^T$ is the unit normal vector pointing outside the computational domain. Expanding in Taylor's series the axial velocity defect expression with respect to the entropy variation $\Delta s = s - s_\infty$ and taking into account for second order terms at most, the entropy drag expression is obtained:

$$D_{\Delta s} = -V_\infty \int_{\mathcal{V}} \boldsymbol{\nabla} \cdot [\rho g(\Delta s) V] \, \mathrm{dV} \,, \tag{B.2}$$

where $\mathcal{V}$ is the flow domain and

$$g(\Delta s) = f_{s1}\left(\frac{\Delta s}{R}\right) + f_{s2}\left(\frac{\Delta s}{R}\right)^2 \,, \tag{B.3}$$

with R the gas constant and the coefficients f_{s1} and f_{s2} given by

$$f_{s1} = -\frac{1}{\gamma M_\infty^2} \quad ; \quad f_{s2} = -\frac{1+(\gamma-1)\,M_\infty^2}{2\gamma^2 M_\infty^4}\,. \tag{B.4}$$

(M_∞ is the free-stream Mach number and γ is the ratio of specific heats).

The entropy drag takes into account for the contributions associated with irreversible processes: viscous and wave drag. The domain $\mathcal{V}$ can be decomposed as $\mathcal{V} = \mathcal{V}_v \bigcup \mathcal{V}_w \bigcup \mathcal{V}_{sp}$, where $\mathcal{V}_v$ is the boundary layer and the wake region, $\mathcal{V}_w$ the shock wave region, and $\mathcal{V}_{sp}$ the remaining part of the flow field. Therefore, the entropy drag can be decomposed in three components:

$$\begin{gathered} D_v = V_\infty \int_{\mathcal{V}_v} \boldsymbol{\nabla}\cdot(\rho g V)\,\mathrm{dV} \quad ; \quad D_w = V_\infty \int_{\mathcal{V}_w} \boldsymbol{\nabla}\cdot(\rho g V)\,\mathrm{dV} \quad ; \\ D_{sp} = V_\infty \int_{\mathcal{V}_{sp}} \boldsymbol{\nabla}\cdot(\rho g V)\,\mathrm{dV}\,. \end{gathered} \tag{B.5}$$

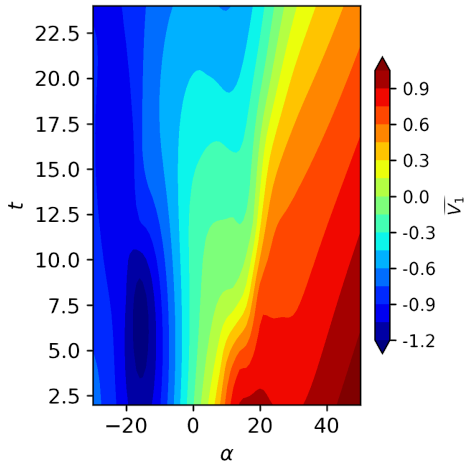
D_v is the viscous drag, D_w is the wave drag and D_{sp} is the spurious drag component. The identification of D_{sp}, introduced by the numerical dissipation, gives chance to improve the accuracy of the drag calculation, in particular for the coarser grids. It is clear that the breakdown method relies on a proper selection of the three domains $\mathcal{V}_v$, $\mathcal{V}_w$ and $\mathcal{V}_{sp}$.

C

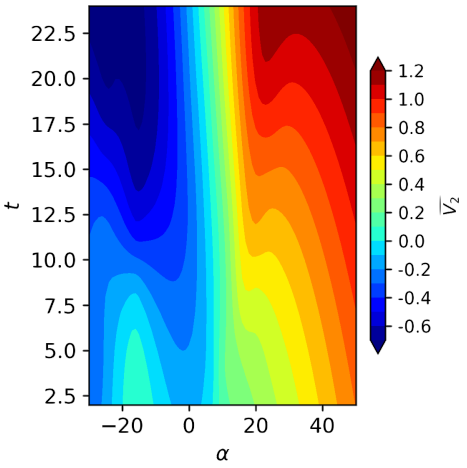
Decoder as generative model

The online deployment of the adopted model, consists in considering only the decoder as generative network. After training the autoencoder to reproduce the input itself, the latent variables are mapped on the database parameters using Radial Basis Functions (RBF) with a Thin-Plate-Spline (TPS) kernel [118] (an example is reported in Figure C.0.1). Using these maps it is possible to determine the latent variables for a new case through interpolation or extrapolation within the latent space (as reported in Figure C.0.2). By providing the new latent variables as input to the trained decoder, it is possible to generate new unseen flow fields. With this approach the encoder serves as a feature extractor, identifying the most relevant features. The decoder, on the other hand, is the actual generative network for the aerodynamic field (for instance around NACA airfoils in this specific case), using the features extracted by the encoder.

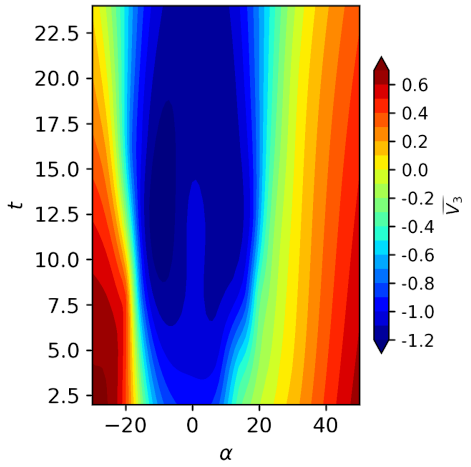
The workflow of this procedure is described in Figure C.0.3.



(a) First latent variable



(b) Second latent variable



(c) Third latent variable

Figure C.0.1: Latent space mapping.

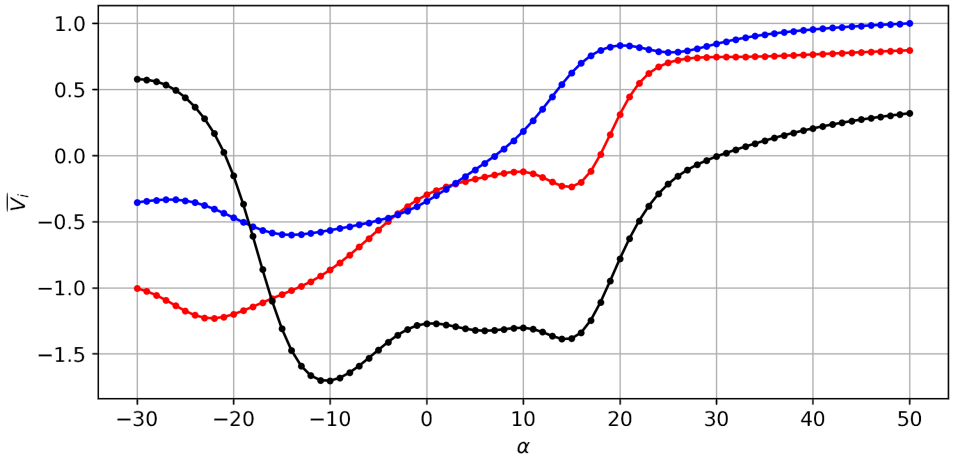


Figure C.0.2: Latent variables, scaled with respect to the maximum value, for the NACA 0012 as function of α extracted by the latent space mappings. Red: V_1; blue: V_2 and black: V_3.

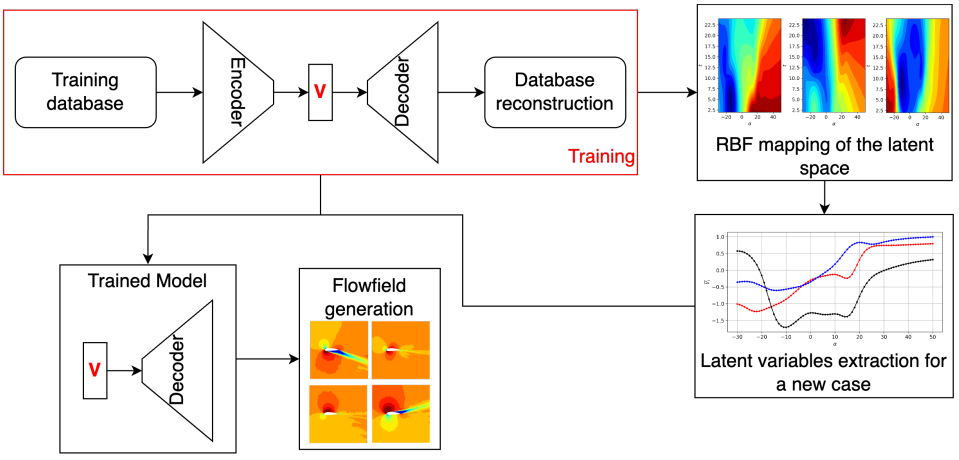


Figure C.0.3: Workflow of the flow field generation procedure.

D

Uncertainty Quantification: additional results

D.1 AE predictions

For the seek for clarity, in the UQ chapter, the prediction fields were not reported. It is useful to show that the field predictions are accurate and in agreement with the ground truth (RANS simulations). In particular, the local Mach number contour is here reported for different angles of attack. Figures D.1.1 to D.1.3 show the comparison between the ground truth obtained with RANS and the flow generated by the decoder. The local Mach number is computed using the three variables generated by the decoder: p, u and v. Considering the free-stream Mach number $M_\infty = 0.15$, it is possible to neglect the density variations due to the compressibility effects and compute the local Mach number using a constant $\rho \approx \rho_\infty$. The percentage error and standard deviation are computed with respect to the free-stream Mach number: $M_\infty = 0.15$.

Figures D.1.1 and D.1.2 are relative to the linear part of the phenomenon. In particular, $\alpha = -5°$ is in the region of the gap of data (previously discussed in section 4), therefore, even if the error is small, it is possible to see more uncertainty in the prediction. On the contrary, for $\alpha = 8°$ the error is negligible, and the model is confident in this prediction.

More interesting is the result of Figure D.1.3 at $\alpha = 22°$. This is in the non-linear part of the phenomenon (after the aerodynamic stall). The error present in the wake of the airfoil is correctly detected by the higer uncertainty provided by the model in that region. As already discussed in the dedicated section 4, this result is very promising, because it enables us to establish

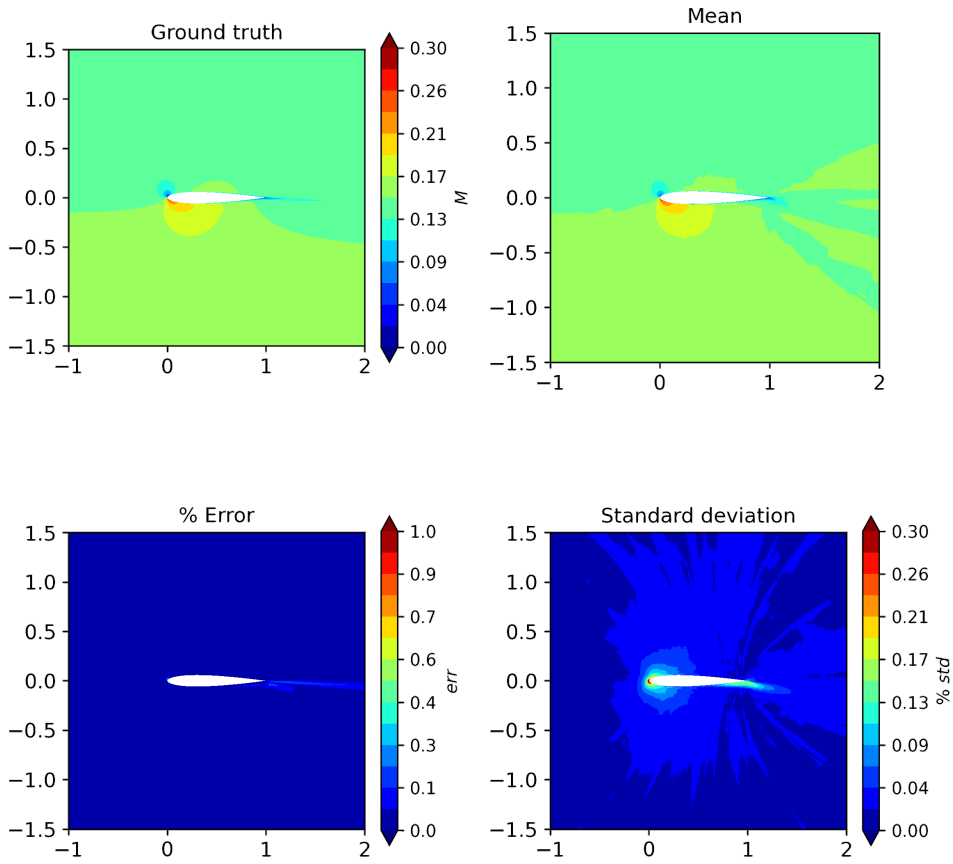


Figure D.1.1: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$, $a = -5°$. Ground truth obtained with RANS (top-left) and mean prediction obtained using the ensemble decoder (top-right). Percentage discrepancy between AE prediction and the ground truth (bottom-left) and standard deviation of AE ensemble (bottom-right).

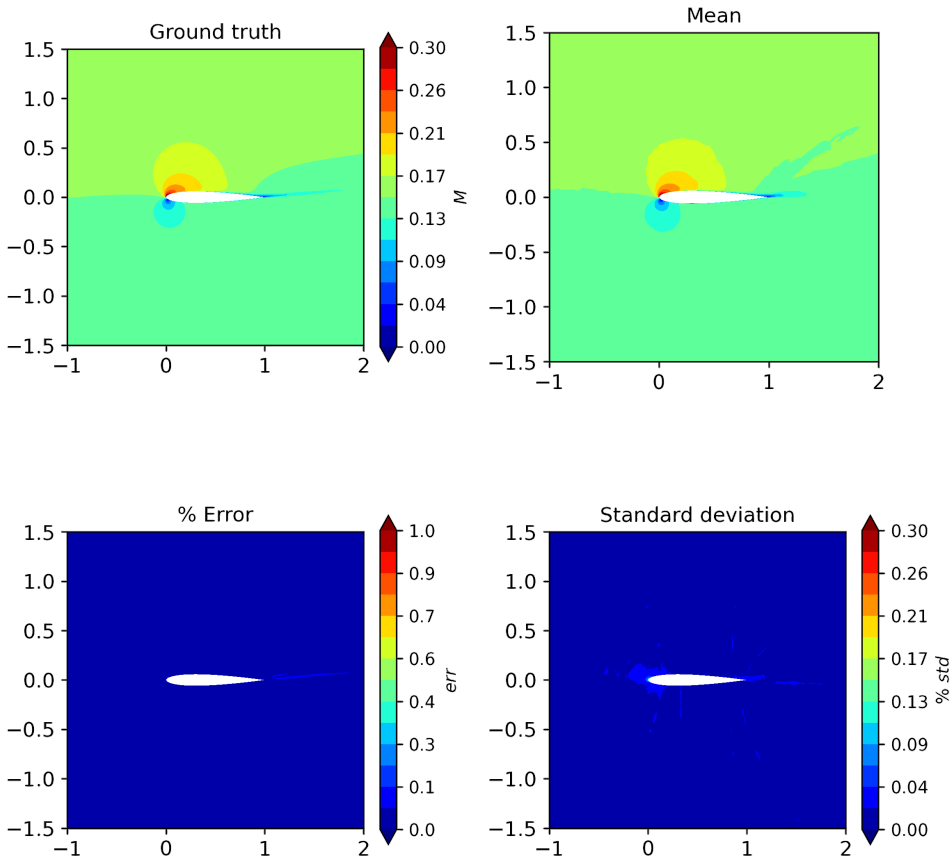


Figure D.1.2: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$, $a = 8°$. Ground truth obtained with RANS (top-left) and mean prediction obtained using the ensemble decoder (top-right). Percentage discrepancy between AE prediction and the ground truth (bottom-left) and standard deviation of AE ensemble (bottom-right).

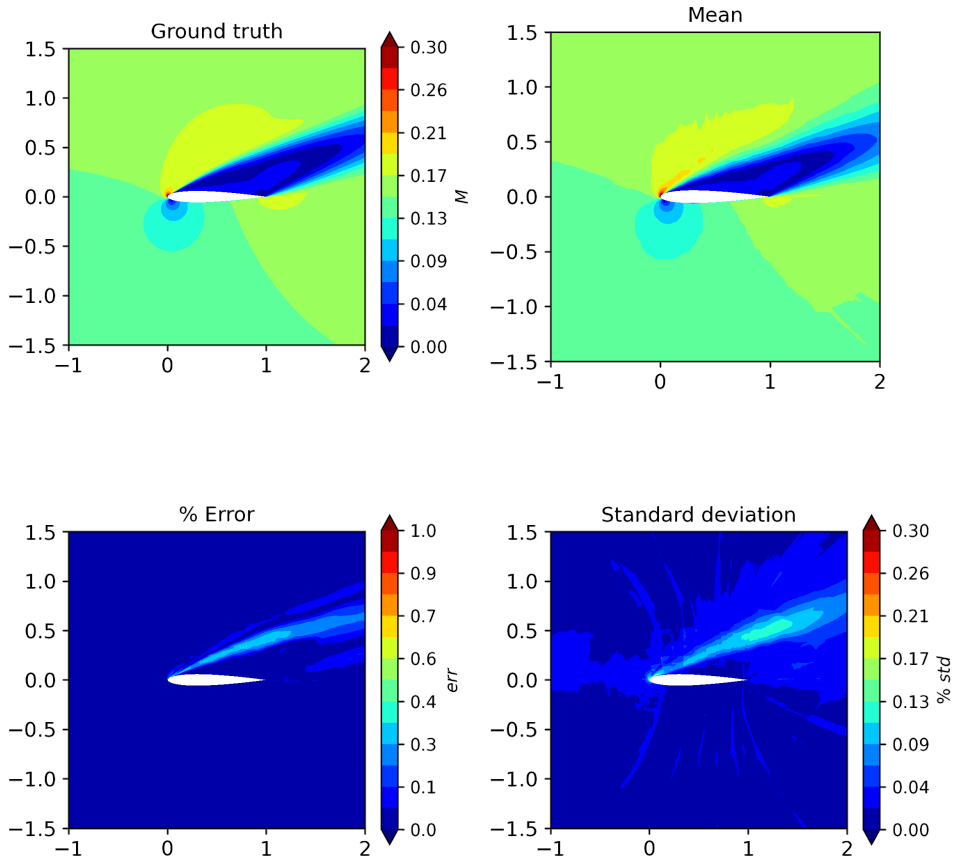


Figure D.1.3: NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$, $a = 22°$. Ground truth obtained with RANS (top-left) and mean prediction obtained using the ensemble decoder (top-right). Percentage discrepancy between AE prediction and the ground truth (bottom-left) and standard deviation of AE ensemble (bottom-right).

that the ensemble standard deviation (which does not require knowledge of the ground truth) is a reasonable proxy for the actual error.

D.2 GPR predictions

Gaussian process regression (GPR) [110] was adopted to reconstruct the flowfield based on the database parameters, namely airfoil thickness and angle of attack. The selection of the kernel function plays a pivotal role in GPR, essentially representing prior knowledge of the phenomenon. Alongside the kernel function, specific hyperparameters, such as the characteristic length scale of the phenomenon, are also essential components. In the present study, the selection of the kernel function is not straightforward, given that GPR is tasked with predicting pressure and velocity components across the entire flowfield. Consequently, we have experimented with three different kernels, and the associated hyperparameters have been chosen through an optimization algorithm (the L-BFGS-B algorithm described in [119]). The optimizer identifies the kernel's parameters that maximize the log-marginal likelihood.

After fitting the GPR, the lift coefficient (quantity of interest) is computed from the flow predictions.

The tested kernel are:

- Radial Basis Function (RBF) in Fig. D.2.1. The kernel is defined as:

$$K^{RBF}(x, x') = \exp\left(-\frac{||x - x'||^2}{2l^2}\right) \tag{D.1}$$

  where $||x - x'||$ is the Euclidean distance between the input points x and x' and l is the length scale: the hyperparameter which controls the smoothness of the function;

- RBF + dot product in Fig. D.2.2. The dot product kernel is defined as:

$$K^{DP}(x, x') = \sigma_0^2 + x \cdot x' \tag{D.2}$$

  where σ_0^2 is an hyperparameter;

- RBF + dot product + white noise in Fig. D.2.3. The white noise kernel is defined as:

$$K^{WN}(x, x') = \begin{cases} nl & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases} \tag{D.3}$$

  where nl is the noise level, which is an hyperparameter selected by the optimizer.

The results reported in Figg. D.2.1-D.2.3 are indicative of the effect of the sensitivity of GPRs to the choice of the kernel and optimization parameters selected. A detailed sensitivity study is outside the scope of the present work, but the results are indicative of predictions that range

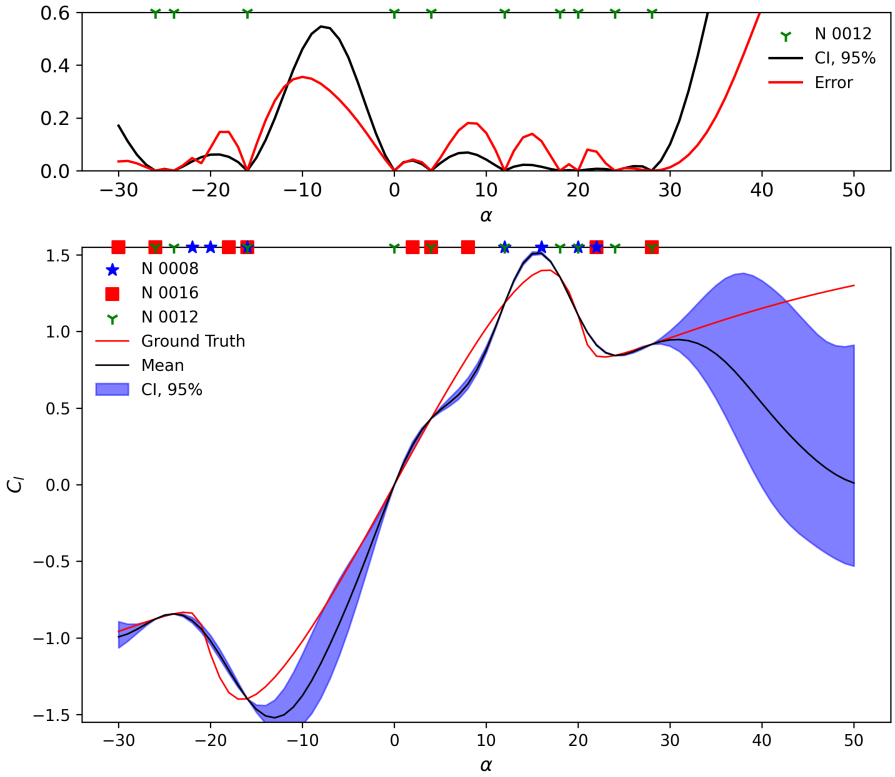


Figure D.2.1: Kernel: RBF. NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$.

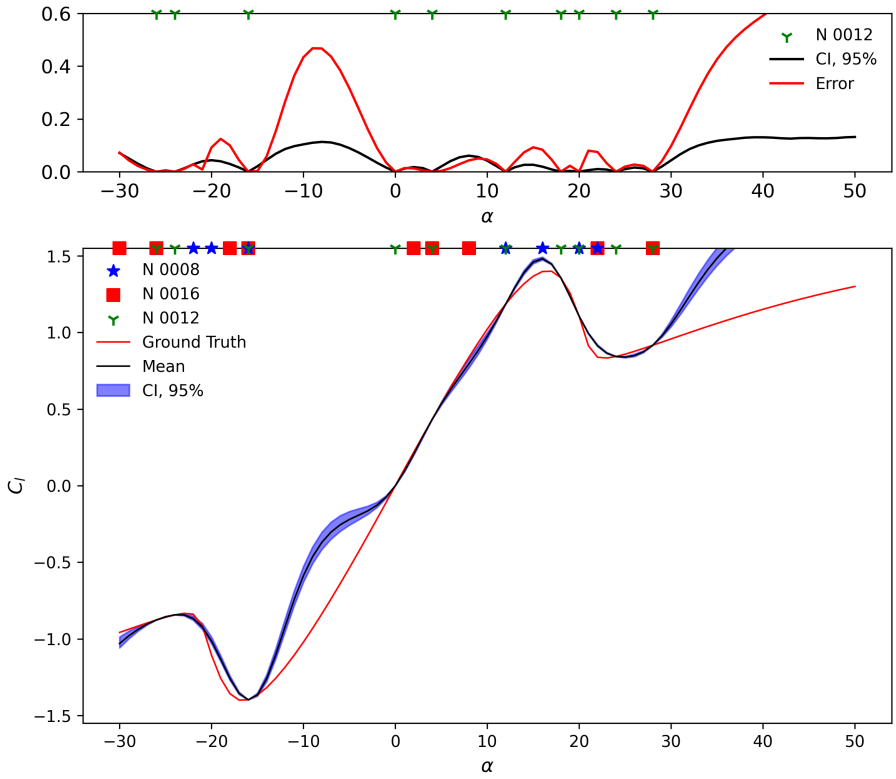


Figure D.2.2: Kernel: RBF + Dot Product. NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$.

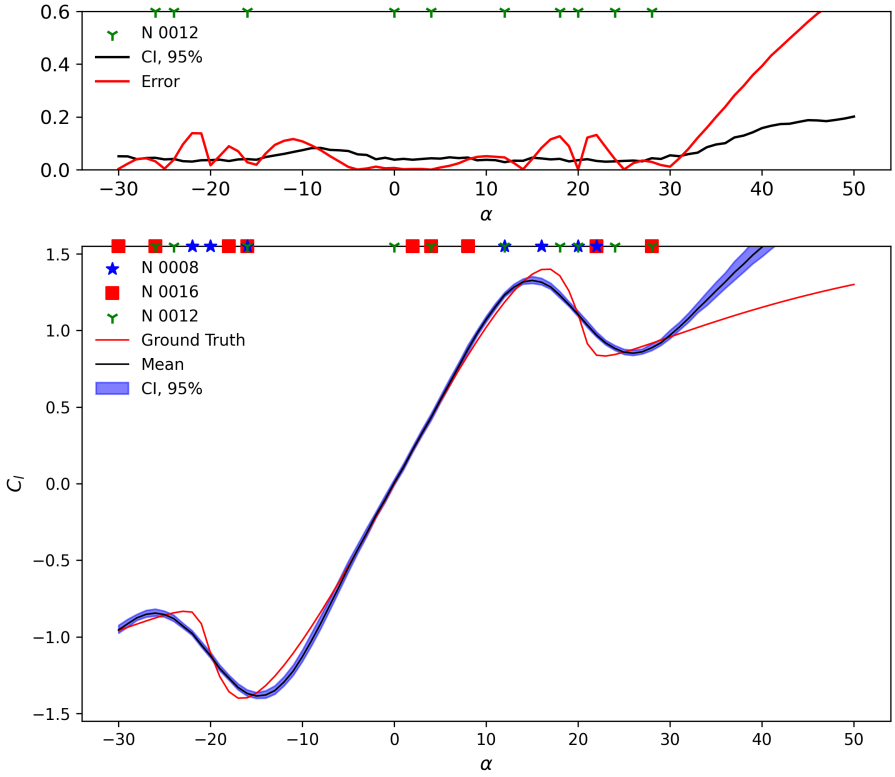


Figure D.2.3: Kernel: RBF + Dot Product + White Noise. NACA 0012, $Re_\infty = 6.0 \times 10^6$, $M_\infty = 0.15$.

from extremely conservative (Fig. D.2.1) to overly confident (Fig. D.2.3). More importantly it is clear from the results presented here (and consistently to what reported in section 4.6) that there is no obvious correlation between the ensemble variability and the prediction error.

References

[1] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020. doi: 10.1146/annurev-fluid-010719-060214.

[2] Andreas C. Müller and Sarah Guido. *Introduction to machine learning with Python : a guide for data scientists*. O'Reilly Media, Inc Sebastopol, CA, 2017. ISBN 9781449369903.

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[4] Haecheon Choi and Parviz Moin. Grid-point requirements for large eddy simulation: Chapman's estimates revisited. *Physics of Fluids*, 24(1):011702, 01 2012. ISSN 1070-6631. doi: 10.1063/1.3676783.

[5] Jeffrey P Slotnick, Abdollah Khodadoust, Juan Alonso, David Darmofal, William Gropp, Elizabeth Lurie, and Dimitri J Mavriplis. Cfd vision 2030 study: A path to revolutionary computational aerosciences. Technical Report NASA/CR-2014-218178, NASA, 2014.

[6] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000. doi: https://doi.org/10.1017/CBO9780511840531.

[7] A. M. Turing. Intelligent machinery. 1948. URL https://www.npl.co.uk/getattachment/about-us/History/Famous-faces/Alan-Turing/80916595-Intelligent-Machinery.pdf.

[8] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. ISSN 00264423, 14602113. URL http://www.jstor.org/stable/2251299.

[9] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

[10] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65, 6:386–408, 1958. doi: 10.1037/h0042519.

[11] M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[12] James Lighthill. Artificial intelligence: A general survey. Report 41, Her Majesty's Stationery Office, 1973. URL https://www.chilton-computing.org.uk/inf/literature/reports/lighthill_report/p001.htm.

[13] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160, 1976. doi: 10.1007/BF01931367.

[14] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8.

[16] Walter Rudin. *Real and Complex Analysis, 3rd Ed.* McGraw-Hill, Inc., USA, 1987. ISBN 0070542341.

[17] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995. doi: 10.1109/ICDAR.1995.598994.

[18] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995. ISSN 1573-0565. doi: 10.1007/BF00994018.

[19] H.T. Siegelmann and E.D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, 1995. ISSN 0022-0000. doi: 10.1006/jcss.1995.1013.

[20] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964.

[21] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503: 92–108, 2022. ISSN 0925-2312. doi: 10.1016/j.neucom.2022.06.111.

[22] Kookjin Lee and Kevin T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404: 108973, 2020. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2019.108973.

[23] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952. doi: 10.1214/aoms/1177729392.

[24] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*, pages 351–368. MIT Press, Cambridge, 2012. ISBN 978-0-262-01646-9.

[25] Andrew Pollard, Luciano Castillo, Luminita Danaila, and Mark N. Glauser. *Whither turbulence and big data in the 21st century?* Springer International Publishing, August 2016. ISBN 9783319412153. doi: 10.1007/978-3-319-41217-7. Publisher Copyright: © Springer International Publishing Switzerland 2017. All rights reserved.

[26] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019. doi: 10.1146/annurev-fluid-010518-040547.

[27] Brendan Tracey, Karthik Duraisamy, and Juan J. Alonso. A machine learning strategy to assist turbulence model development. *AIAA SciTech*, 2015. doi: 10.2514/6.2015-1287. 53rd AIAA Aerospace Sciences Meeting, 5-9 January 2015, Kissimmee, Florida.

[28] Michele Milano and Petros Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002. ISSN 0021-9991. doi: 10.1006/jcph.2002.7146.

[29] Karthik Duraisamy, Ze Jia Zhang, and Anand Pratap Singh. New approaches in turbulence and transition modeling using data-driven techniques. *AIAA SciTech*, 2015. doi: https://doi.org/10.2514/6.2015-1284. 53rd AIAA Aerospace Sciences Meeting, 5-9 January 2015, Kissimmee, Florida.

[30] Xinghui Yan, Jihong Zhu, Minchi Kuang, and Xiangyang Wang. Aerodynamic shape optimization using a novel optimizer based on machine learning techniques. *Aerospace Science and Technology*, 86:826–835, 2019. doi: https://doi.org/10.1016/j.ast.2019.02.003.

[31] Jichao Li, Mengqi Zhang, Joaquim R. R. A. Martins, and Chang Shu. Efficient aerodynamic shape optimization with deep-learning-based geometric filtering. *AIAA Journal*, 58(10), 2020. doi: 10.2514/1.J059254.

[32] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.09.003.

[33] C. L. Teo, Kah Bin Lim, Geok-Soon Hong, and M.H.T. Yeo. A neural net approach in analyzing photograph in piv. *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1535–1538 vol.3, 1991. URL https://api.semanticscholar.org/CorpusID:61609966.

[34] I. Grant and X. Pan. An investigation of the performance of multi layer, neural networks applied to the analysis of piv images. *Experiments in Fluids*, 19:159–166, 1995. doi: doi.org/10.1007/BF00189704.

[35] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics informed learning machine, 2021. US Patent 10,963,540.

[36] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: a review. *Acta Mechanica Sinica*, 37:1727–1738, 2021. ISSN 1614-3116. doi: 10.1007/s10409-021-01148-1.

[37] Jichao Li, Xiaosong Du, and Joaquim R.R.A. Martins. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences*, 134:100849, 2022. ISSN 0376-0421. doi: 10.1016/j.paerosci.2022.100849.

[38] C. Vignon, J. Rabault, and R. Vinuesa. Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions. *Physics of Fluids*, 35(3): 031301, 03 2023. ISSN 1070-6631. doi: 10.1063/5.0143913.

[39] Ricardo Vinuesa, Steven L. Brunton, and Beverley J. McKeon. The transformative potential of machine learning for experiments in fluid mechanics. *Nature Reviews Physics*, 5: 536–545, 2023. ISSN 2522-5820. doi: 10.1038/s42254-023-00622-y.

[40] Stefano Discetti and Yingzheng Liu. Machine learning for flow field measurements: a perspective. *Measurement Science and Technology*, 34(021001), 2022.

[41] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016. doi: 10.1017/jfm.2016.615.

[42] Octavi Obiols-Sales, Abhinav Vishnu, Nicholas Malaya, and Aparna Chandramowliswharan. Cfdnet: A deep learning-based accelerator for fluid simulations. In *Proceedings of the 34th ACM International Conference on Supercomputing*, ICS '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379830. doi: 10.1145/3392717.3392772. URL https://doi.org/10.1145/3392717.3392772.

[43] Filipe de Avila Belbute-Peres, Thomas D. Economon, and J. Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[44] Thomas D. Economon, Francisco Palacios, Sean R. Copeland, Trent W. Lukaczyk, and Juan J. Alonso. Su2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846, 2016. doi: https://doi.org/10.2514/1.J053813.

[45] Yao Zhang, Woong Je Sung, and Dimitri N. Mavris. *Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient*. 2018. doi: 10.2514/6.2018-1903. URL https://arc.aiaa.org/doi/abs/10.2514/6.2018-1903.

[46] Vinothkumar Sekar, Qinghua Jiang, Chang Shu, and Boo Cheong Khoo. Fast flow field prediction over airfoils using deep learning approach. *Physics of Fluids*, 31(5):057103, 05 2019. ISSN 1070-6631. doi: 10.1063/1.5094943.

[47] Haizhou Wu, Xuejun Liu, Wei An, Songcan Chen, and Hongqiang Lyu. A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils. *Computers & Fluids*, 198:104393, 2020. ISSN 0045-7930. doi: 10.1016/j.compfluid.2019.104393.

[48] Lionel Agostini. Exploration and prediction of fluid dynamical systems using auto-encoder technology. *Physics of Fluids*, 32(6):067103, 2020. doi: https://doi.org/10.1063/5.0012906.

[49] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019. doi: 10.1007/s00466-019-01740-0.

[50] Sangseung Lee and Donghyun You. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, 879:217–254, 2019. doi: 10.1017/jfm.2019.700.

[51] Kaustubh Tangsali, Vinayak R. Krishnamurthy, and Zohaib Hasnain. Generalizability of Convolutional Encoder–Decoder Networks for Aerodynamic Flow-Field Prediction Across Geometric and Physical-Fluidic Variations. *Journal of Mechanical Design*, 143(5), 11 2020. ISSN 1050-0472. doi: https://doi.org/10.1115/1.4048221.

[52] Yu-Eop Kang, Sunwoong Yang, and Kwanjung Yee. Physics-aware reduced-order modeling of transonic flow via β-variational autoencoder. *Physics of Fluids*, 34(7):076103, 07 2022. ISSN 1070-6631. doi: 10.1063/5.0097740.

[53] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8237. doi: 10.1561/2200000056.

[54] Lucas Pinheiro Cinelli, Matheus Araújo Marins, Eduardo Antúnio Barros da Silva, and Sérgio Lima Netto. *Variational Autoencoder*, pages 111–149. Springer International Publishing, 2021. ISBN 978-3-030-70679-1. doi: 10.1007/978-3-030-70679-1_5.

[55] Runze Li, Yufei Zhang, and Haixin Chen. Physically interpretable feature learning of supercritical airfoils based on variational autoencoders. *AIAA Journal*, 60(11):6168–6182, 2022. doi: https://doi.org/10.2514/1.J061673.

[56] M. Lanzetta, B. Mele, and R Tognaccini. Advances in aerodynamic drag extraction by far field methods. *Journal of Aircraft*, 52(6), 2014. doi: https://doi.org/10.2514/1.C033095.

[57] Jared L. Callaham, James V. Koch, Bingni W. Brunton, J. Nathan Kutz, and Steven L. Brunton. Learning dominant physical processes with data-driven balance models. *Nature Communications*, 12, 2021. doi: https://doi.org/10.1038/s41467-021-21331-z.

[58] Bryan E. Kaiser, Juan A. Saenz, Maike Sonnewald, and Daniel Livescu. Automated identification of dominant physical processes. *Engineering Applications of Artificial Intelligence*, 116:105496, 2022. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2022.105496.

[59] Maike Sonnewald, Carl Wunsch, and Patrick Heimbach. Unsupervised learning reveals geography of global ocean dynamical regions. *Earth and Space Science*, 6(5):784–794, 2019. doi: https://doi.org/10.1029/2018EA000519.

[60] Eva Patel and Dharmender Singh Kushwaha. Clustering cloud workloads: K-means vs gaussian mixture model. *Procedia Computer Science*, 171:158–167, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.04.017. Third International Conference on Computing and Network Communications (CoCoNet'19).

[61] S. P. Lloyd. Least squares quantization in pcm. Technical Report RR-5497, Bell Lab, 1957.

[62] J. B MacQueen. Some methods for classification and analysis of multivariate observations. In *L. M. Le Cam & J. Neyman (Eds.), Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 281–297, 1967.

[63] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[64] Jinhee Jeong and Fazle Hussain. On the identification of a vortex. *Journal of Fluid Mechanics*, 285:69–94, 1995. doi: 10.1017/S0022112095000462.

[65] Z. Wu, J. Lee, C. Meneveau, and T. Zaki. Application of a self-organizing map to identify the turbulent-boundary-layer interface in a transitional flow. *Phisical Review Fluids*, 4 (023902), 2019.

[66] L Qiao, XL He, Y Sun, JQ Bai, and L Li. Far-field drag decomposition using hybrid formulas and vorticity based area sensors. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 235(11):1411–1426, 2021. doi: https://doi.org/10.1177/0954410020973904.

[67] David Lovely and Robert Haimes. Shock detection from computational fluid dynamics results. In *14th Computational Fluid Dynamics Conference*, pages 296–304, 1999. doi: https://doi.org/10.2514/6.1999-3285.

[68] L. Paparone and R. Tognaccini. Computational fluid dynamics-based drag prediction and decomposition. *AIAA Journal*, 41(9):1647–1657, 2003. doi: https://doi.org/10.2514/2.7300.

[69] F Ducros, V Ferrand, F Nicoud, C Weber, D Darracq, C Gacherieu, and T Poinsot. Large-eddy simulation of the shock/turbulence interaction. *Journal of Computational Physics*, 152(2):517–549, 1999. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1999.6238.

[70] Ziniu Wu, Yizhe Xu, Wenbin Wang, and Ruifeng Hu. Review of shock wave detection method in cfd post-processing. *Chinese Journal of Aeronautics*, 26(3):501–513, 2013. ISSN 1000-9361. doi: https://doi.org/10.1016/j.cja.2013.05.001.

[71] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[72] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[73] Pierce L. Hart and Sven Schmitz. Partial pressure field for airfoil wave drag. *AIAA Journal*, 60(10):5791–5804, 2022. doi: https://doi.org/10.2514/1.J061690.

[74] John Vassberg. A unified baseline grid about the common research model wing/body for the fifth aiaa cfd drag prediction workshop (invited). In *29th AIAA Applied Aerodynamics Conference*. doi: 10.2514/6.2011-3508.

[75] Anthony J. Sclafani, John C. Vassberg, Chad Winkler, Andrew J. Dorgan, Mori Mani, Michael E. Olsen, and James G. Coder. Analysis of the common research model using structured and unstructured meshes. *Journal of Aircraft*, 51(4):1223–1243, 2014. doi: https://doi.org/10.2514/1.C032411.

[76] Pierce Hart, Christopher J. Axten, Mark D. Maughmer, and Sven Schmitz. Drag decomposition of full aircraft configurations using partial-pressure fields. *AIAA AVIATION 2022 Forum*, 2022. doi: https://doi.org/10.2514/6.2022-3885.

[77] Camille Fournis, Didier Bailly, and Renato Tognaccini. An invariant vortex-force theory related to classical far-field analyses in transonic flows. *AIAA AVIATION 2021 FORUM*, 2021. doi: https://doi.org/10.2514/6.2021-2554.

[78] Hamidreza Eivazi, Soledad Le Clainche, Sergio Hoyas, and Ricardo Vinuesa. Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows. *Expert Systems with Applications*, 202:117038, 2022. ISSN 0957-4174. doi: 10.1016/j.eswa.2022.117038.

[79] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

[80] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019. doi: 10.5555/3454287.3455008.

[81] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 481–490, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939738.

[82] Sangseung Lee and Donghyun You. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, 879:217–254, sep 2019. doi: 10.1017/jfm.2019.700.

[83] Alasdair Newson, Andrés Almansa, Yann Gousseau, and Saïd Ladjal. Processing simple geometric attributes with autoencoders. *Journal of Mathematical Imaging and Vision*, 62 (3):293–312, 2020. doi: https://doi.org/10.1007/s10851-019-00924-w.

[84] I. H. A. Abbott and Albert E. von Doenhoff. *Theory of Wing Sections*. Dover Publications Inc., 1959.

[85] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[86] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979. ISSN 00401706.

[87] Yong-Dao Zhou, Kai-Tai Fang, and Jian-Hui Ning. Mixture discrepancy for quasi-random point sets. *Journal of Complexity*, 29(3):283–301, 2013. ISSN 0885-064X. doi: 10.1016/j.jco.2012.11.006.

[88] K. W. Mcalister, L. W. Carr, and W. J. Mccroskey. Dynamic stall experiments on the naca 0012 airfoil. Technical Report 19780009057, NASA, 1978.

[89] Marco Raiola, Stefano Discetti, and Andrea Ianiro. Data-driven identification of unsteady-aerodynamics phenomena in flapping airfoils. *Experimental Thermal and Fluid Science*, 124:110234, 2021. ISSN 0894-1777. doi: 10.1016/j.expthermflusci.2020.110234.

[90] Kobra Gharali and David A. Johnson. Dynamic stall simulation of a pitching airfoil under unsteady freestream velocity. *Journal of Fluids and Structures*, 42:228–244, 2013. ISSN 0889-9746. doi: 10.1016/j.jfluidstructs.2013.05.005.

[91] T. LEE and P. GERONTAKOS. Investigation of flow over an oscillating airfoil. *Journal of Fluid Mechanics*, 512:313–341, 2004. doi: 10.1017/S0022112004009851.

[92] J. G. Leishman and T. S. Beddoes. A semi-empirical model for dynamic stall. *Journal of the American Helicopter Society*, 34(3):3–17, 1989. doi: 10.4050/JAHS.34.3.3.

[93] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. ISSN 1566-2535. doi: 10.1016/j.inffus.2021.05.008.

[94] H. M. Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE Access*, 6:36218–36234, 2018. doi: 10.1109/ACCESS.2018.2836917.

[95] Apostolos F. Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023. ISSN 0021-9991. doi: 10.1016/j.jcp.2022.111902.

[96] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8237. doi: https://doi.org/10.1561/2200000056.

[97] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[98] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*, 2017.

[99] Mingliang Liu, Dario Grana, and Leandro Passos de Figueiredo. Uncertainty quantification in stochastic inversion with dimensionality reduction using variational autoencoder. *Geophysics*, 87(2):M43–M58, 12 2021. ISSN 0016-8033. doi: 10.1190/geo2021-0138.1.

[100] Bang Xiang Yong and Alexandra Brintrup. Bayesian autoencoders with uncertainty quantification: Towards trustworthy anomaly detection. *Expert Systems with Applications*, 209:118196, 2022. ISSN 0957-4174. doi: 10.1016/j.eswa.2022.118196.

[101] Lorenzo Perini, Vincent Vercruyssen, and Jesse Davis. Quantifying the confidence of anomaly detectors in their example-wise predictions. In Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 227–243, Cham, 2021. Springer International Publishing. ISBN 978-3-030-67664-3.

[102] Ethan Pickering, Stephen Guth, George Em Karniadakis, and Themistoklis P. Sapsis. Discovering and forecasting extreme events via active learning in neural operators. *Nature Computational Science*, 2:823–833, 2022. doi: 10.1038/s43588-022-00376-0.

[103] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. doi: 10.1109/34.58871.

[104] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861.

[105] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[106] Michael R. Chernick, Wenceslao González-Manteiga, Rosa M. Crujeiras, and Erniel B. Barrios. *Bootstrap Methods*, pages 169–174. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_150.

[107] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929 – 1958, 2014.

[108] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40:33–51, 1975. ISSN 1860-0980. doi: 10.1007/BF02291478.

[109] W.J. Krzanowski. *Principles of Multivariate Analysis: A User's Perspective*. Oxford Statistical Science Series, Oxford University Press, 2000. ISBN 9780198507086.

[110] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005. ISBN 9780262256834. doi: 10.7551/mitpress/3206.001.0001.

[111] Aashwin Ananda Mishra, Jayant Mukhopadhaya, Gianluca Iaccarino, and Juan Alonso. Uncertainty estimation module for turbulence model predictions in su2. *AIAA Journal*, 57(3):1066–1077, 2019. doi: 10.2514/1.J057187.

[112] D. Destarac and J. van der Vooren. Drag/thrust analysis of jet-propelled transonic transport aircraft; definition of physical drag components. *Aerospace Science and Technology*, 8 (6):545 – 556, 2004. ISSN 1270-9638. doi: https://doi.org/10.1016/j.ast.2004.03.004.

[113] J.-Z. Wu, X.-Y. Lu, and L.-X. Zhuang. Integral force acting on a body due to local flow structures. *Journal of Fluid Mechanics*, 576:265–286, April 2007.

[114] B. Mele and R. Tognaccini. Aerodynamic force by lamb vector integrals in compressible flow. *Physics of Fluids*, 26, May 2014.

[115] L. Q. Liu, J. Z. Wu, W. D. Su, and L. L. Kang. Lift and drag in three-dimensional steady viscous and compressible flow. *Physics of Fluids*, 29:116105–1–13, 2017. doi: https://doi.org/10.1063/1.4989747.

[116] L. Q. Liu. *Unified Theoretical Foundations of Lift and Drag in Viscous and Compressible External Flows*. PhD thesis, Peking University, Beijing, China, 2018.

[117] Sven Schmitz. Drag decomposition using partial-pressure fields in the compressible navier–stokes equations. *AIAA Journal*, 57(5):2030–2038, 2019. doi: https://doi.org/10.2514/1.J057701.

[118] G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1990. ISBN 9780898712445.

[119] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. 23(4):550–560, dec 1997. ISSN 0098-3500. doi: 10.1145/279232.279236.

Published Excerpts of Present Dissertation

JOURNAL ARTICLES

- E. Saetta and R. Tognaccini. *"Identification of Flowfield Regions by Machine Learning"*, AIAA Journal, 61:4, 1503-1518 (2023). DOI: https://doi.org/10.2514/1.J061907.
- E. Saetta, R. Tognaccini and G. Iaccarino *"Machine Learning to Predict Aerodynamic Stall"*, International Journal of Computational Fluid Dynamics, 36:7, 641-654 (2023). DOI: https://doi.org/10.1080/10618562.2023.2171021.
- E. Saetta, R. Tognaccini and G. Iaccarino *"Uncertainty Quantification for Machine Learning Aerodynamic Predictions"*, Journal of Computational Physics (submitted).

CONFERENCE PAPERS*

- E. Saetta and R. Tognaccini. *"Identification of flow field regions by Machine Learning"*, AIAA 2022-0457, AIAA SCITECH 2022 Forum, 3-7 January 2022. DOI: https://doi.org/10.2514/6.2022-0457.
- E. Saetta, R. Tognaccini and G. Iaccarino. *"AbbottAE: An Autoencoder for Airfoil Aerodynamics"*, AIAA AVIATION 2023 FORUM, 12–16 June 2023, San Diego, CA (USA). DOI: https://doi.org/10.2514/6.2023-4364.

CONFERENCE (BOOK OF ABSTRACTS ONLY)

- E. Saetta, R. Tognaccini and G. Iaccarino. *"Can autoencoders predict airfoil stall?"*, IUTAM Symposium on Data-driven modeling and optimization in fluid mechanics, 15-17 June 2022, Aarhus C, Denmark. URL: https://conferences.au.dk/iutam.
- E. Saetta, R. Tognaccini and G. Iaccarino. *"Towards Predicting Dynamic Stall using Machine Learning Tools"*, 14th European Fluid Mechanics Conference, 13-16 September 2022, Athens, Greece. URL: https://www.efmc14.org/EFMC14_ABSTRACT_BOOK.pdf.
- E. Saetta and R. Tognaccini. *"Towards a "headache-free" flow region selection"*, 57th 3AF International Conference on Applied Aerodynamics (AERO2023), 29-31 March 2023, Bordeaux, France.

*Speaker.

Colophon

This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Arno Pro, designed by Robert Slimbach in the style of book types from the Aldine Press in Venice, and issued by Adobe in 2007. A template, which can be used to format a PhD thesis with this look and feel, has been released under the permissive MIT (X11) license, and can be found online at github.com/suchow/ or from the author at suchow@post.harvard.edu.