aerospace

MDPI

Article

Nonlinear Surrogate Model Design for Aerodynamic Dataset Generation Based on Artificial Neural Networks

Guillermo Suarez [1,*], Emre Özkaya [1], Nicolas R. Gauger [1], Hans-Jörg Steiner [2], Michael Schäfer [2] and David Naumann [2]

[1] Chair for Scientific Computing, University of Kaiserslautern-Landau (RPTU), 67663 Kaiserslautern, Germany; emre.oezkaya@scicomp.uni-kl.de (E.Ö.); nicolas.gauger@scicomp.uni-kl.de (N.R.G.)
[2] Airbus Defense and Space (AD&S), 85077 Manching, Germany; hans-joerg.steiner@airbus.com (H.-J.S.); michael.m.schaefer@airbus.com (M.S.); david.d.naumann@airbus.com (D.N.)
* Correspondence: guillermo.suarez@scicomp.uni-kl.de

Abstract: In this work we construct a surrogate model using artificial neural networks (ANN) to predict the steady-state behavior of an unmanned combat aircraft. We employ various strategies to improve the model's accuracy, including the consideration of design tolerances, creating independent surrogate models for the different flow regimes and encoding non-numeric input features. We also explore alternative machine learning models, albeit they demonstrated a lower reliability than ANNs. Two scenarios are considered for the target variable: one focusing solely on predicting the pitching moment coefficient, and the other incorporating the roll moment coefficient as well. We investigate different methods for handling multiple targets, finding that constructing a single model with multiple outputs consistently outperforms developing separate models for each target variable. Overall, the ANN provides predictions that show excellent agreement with the experimental data, demonstrating its effectiveness and reliability in aerodynamic modeling.

Keywords: artificial neural network; surrogate model; aerodynamics

check for updates

Citation: Suarez, G.; Özkaya, E.; Gauger, N.R.; Steiner, H.-J.; Schäfer, M.; Naumann, D. Nonlinear Surrogate Model Design for Aerodynamic Dataset Generation Based on Artificial Neural Networks. *Aerospace* **2024**, *11*, 607. https://doi.org/10.3390/aerospace11080607

Academic Editor: Sergey Leonov

Received: 29 May 2024
Revised: 12 July 2024
Accepted: 19 July 2024
Published: 24 July 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

When engineers embark on designing a new aircraft or upgrading an existing one, a physical representation of the aircraft's aerodynamic behavior throughout various stages of the flight envelope is fundamental for assessment, optimization, and iterative design. For that purpose, surrogate models serve as a rapid prediction tool, facilitating the computation of aerodynamic properties under diverse flight conditions without the need for extensive measurements.

Traditional aerodynamic surrogate models often rely on multivariate interpolation methods applied to lookup tables. These methods are currently widely used in the aerospace sector and demonstrated to perform well for linear regimes. However, their accuracy diminishes in nonlinear regimes unless the data tables are significantly refined and densified. As a result, a combination of substantial wind tunnel tests, flight experiments, and/or numerical simulations, such as computational fluid dynamics (CFD), are necessary, incurring significant time and economic expenses. Therefore, exploring alternative, more cost-effective and low-risk methodologies becomes imperative to streamline the aircraft design process.

Machine learning (ML) models are capable of handling the complex, nonlinear relationships between input parameters, such as geometry, flow conditions, and angles of attack, and the resulting aerodynamic coefficients. Methods like artificial neural networks (ANN) and ensemble learning (EL) excel at capturing and modeling these intricate dependencies. This capability allows machine learning to provide highly accurate predictions across various conditions. Furthermore, ML models can generalize from training data

to accurately predict new, unseen configurations, which is vital for designing and optimizing new aerospace components where novel shapes and flow conditions frequently arise. For example, Bhatnagar et al. [1] utilized convolutional neural networks to predict the velocity and pressure fields around airfoils. Eivazi et al. [2] are able to solve the Reynolds-averaged Navier–Stokes (RANS) equations for incompressible turbulent flows using physics-informed neural networks. Recently, Zahn et al. [3] showed the applicability of long short-term memory neural networks for the prediction of transonic buffet aerodynamics. Moreover, given the power of modern computers, these models can be trained quickly surpassing the computational time bottleneck from CFD. Once trained, the ML models can rapidly generate predictions for new conditions without requiring additional simulations or tests. This efficiency can streamline the design process and potentially cut production costs in half.

Both ANN and EL methods have already demonstrated effectiveness in constructing surrogate models for aerodynamic applications [4–7]. For instance, Ross et al. [8] successfully created a response surface for multiple aerodynamic coefficients of a fighter aircraft configuration using wind tunnel data. They developed independent neural networks, each with multiple inputs for a specific target variable, and found that this single-output approach outperformed a single model with multiple outputs.

Conversely, Karali et al. [9] predicted the lift, drag, and pitching moment coefficients for an unmanned air vehicle using data generated by a nonlinear lifting line tool. They created a single neural network that predicted the three coefficients simultaneously, demonstrating very good agreement with the reference data. Additionally, Patri and Patnaik [10] utilized random forest and stochastic gradient tree boosting for predicting noise levels around a NACA0012 airfoil, relying on wind tunnel data.

These studies highlight the versatility and robustness of machine learning approaches in aerodynamic modeling, suggesting that different approaches may be suited to different types of aerodynamic data and prediction goals. However, there is a notable gap in the literature regarding comparison studies between different machine learning models.

An important consideration in this context is that machine learning models rely on data to approximate or regress the relationship between specific inputs and target variables. One significant challenge in constructing these models is achieving a balance between generalization and overfitting. The model must accurately predict the target variables while retaining the ability to generalize to new, unseen data [11]. Striking this balance is crucial for making these models applicable to industrial contexts. Particularly in the aircraft design process, it is essential to ensure compliance with design tolerances, a critical aspect that is often overlooked.

This work focuses on constructing a surrogate model to predict the steady-state nonlinear aerodynamic behavior of an unmanned fighter aircraft, leveraging wind tunnel data and machine learning models. We delve into evaluating two distinct data-driven machine-learning approaches to construct a nonlinear surrogate model: artificial neural networks and ensemble learning models. Furthermore, we address challenges such as encoding non-numeric variables and handling the extension of data through different flow regimes. Additionally, we explore the treatment of multiple targets, investigating whether simultaneous prediction or individual models for each target are more accurate. Throughout our design, we emphasize the importance of adhering to design tolerances, ensuring that our models are not only accurate but also practical for real-world applications.

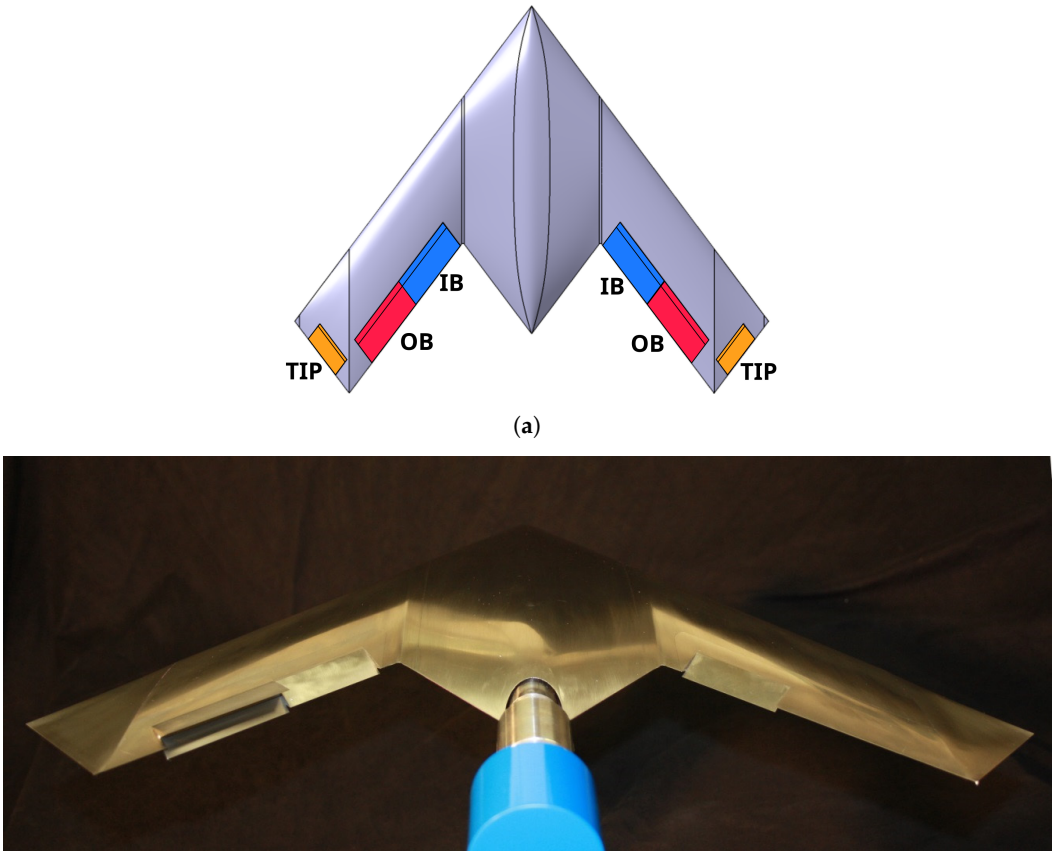
In Section 2 of the paper, the unmanned fighter aircraft used as a test case is presented. Section 3 considers and discusses the machine learning methodologies, which are the focus of the paper. This includes the definition of the models and hyperparameter optimization as well as the steps required to prepare and analyze the data. Then, in Section 4 the results are shown. The discussion including concluding remarks is the subject of Section 5.

2. Test Case Definition

As a test case scenario, our study focuses on predicting the stability and control (S&C) of the Unmanned Combat Air Vehicle DLR-F17, known as SACCON (Stability And Control CONfiguration), alongside the DLR-F19. These two aircraft models share identical planforms, differing only in the functionality of their control surfaces. Therefore, throughout this document, we will collectively refer to both models as SACCON. The model layout depicting the SACCON configuration and its control surfaces is illustrated in Figure 1a. Each wing of the SACCON model features three control surfaces, including elevons and split flaps, designated as follows: TIP: Tip Elevon; OB: Outboard Flap; IB: Inboard Flap. Whenever an L or R is stated in front of one of these acronyms, it denotes the Left or Right wing, respectively. Additionally, it is important to note that the outboard flap has the capability to split symmetrically, as depicted in Figure 1b. In this study, the convention for the upward and downward deflection of control surfaces is as follows: elevons deflected downwards will be indicated with a positive angle, while negative angles will correspond to upwards deflections. Hereafter, we will refer to the aircraft configuration with no control surface deflection as the *baseline configuration*.

As a first step toward achieving a comprehensive understanding and gaining experience in developing a surrogate model for the S&C prediction, we focus on predicting the pitching moment coefficient, C_m. The pitching moment plays a fundamental role in the aircraft overall stability and controllability, influencing its behavior in flight. It represents the aerodynamic force that tends to rotate the aircraft around its lateral axis, causing it to pitch up or down, being essential to determine the maneuverability of an aircraft. As such it is the main driver in the determination of longitudinal static stability. Stability and control issues in the longitudinal axis are, therefore, critical to identify for flight control design, in particular in case of nonlinearities and rapid gradient changes. Such issues may be caused by complex vortex flow phenomena, e.g., vortex burst [12]. Moreover, we will analogously develop a surrogate model for the rolling moment coefficient, C_l. Similar to C_m, it also plays a crucial role in the stability and control of an aircraft, albeit in the lateral direction. For instance, the dependence of C_l on sideslip angle significantly influences the stability of aircraft, particularly at high angles of attack. Consequently, this aspect holds paramount importance, especially in the context of combat aircraft.

In particular, surrogate models are designed to expedite solution processes while maintaining reliability and integrity. Therefore, a certain level of accuracy and fidelity is expected, meaning that the ML algorithm should capture all significant characteristics of the system. The aerodynamics of the SACCON involve intricate flow and solid interactions, characterized by complex and highly nonlinear relationships. This complexity increases significantly when modifying the aircraft configuration due to phenomena like vortex shedding, separation, and reattachment. The developed model must ensure it adequately represents the flow physics within the relevant flight domain, for example, in order to enable an appropriate control law design and load identification. For this purpose, it must demonstrate acceptable accuracy and encompass design tolerances. In this work, we have established three distinct types of tolerances for each moment coefficient: one for the coefficient itself, and others for its derivative with respect to the angle of attack (AoA) and the angle of sideslip (AoS). The design tolerance represents the level of uncertainty typically encountered in wind tunnel testing.



(**a**)

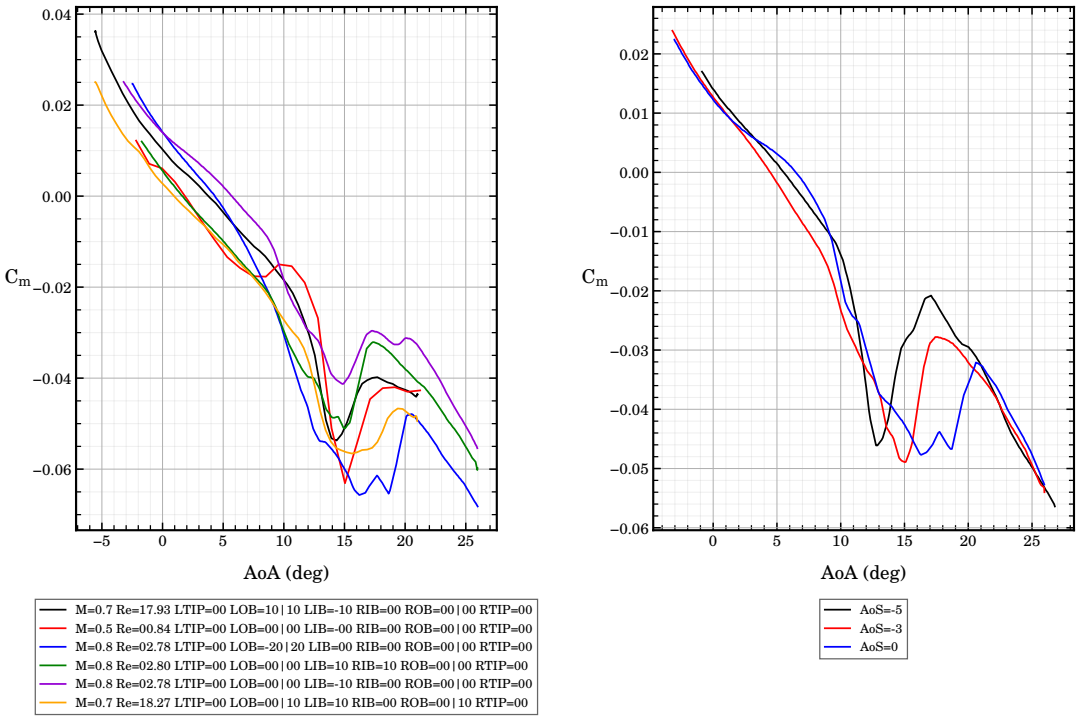
(**b**)

Figure 1. SACCON planform and control surfaces visualization. (**a**) Control surfaces and naming convention. Image adapted from [13], with permission. (**b**) Rear view of the DLR-F17E model with control surfaces on trailing edges, left to right: LTIP (0°), split LOB (20°), LIB (10°), RIB (10°), ROB (0°), RTIP (0°). Picture from [14], with permission.

As part of a collaborative research program within the NATO Research & Technology Organisation different research organizations, such as the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V.) and NASA Langley Research Center has conducted extensive experimental measurements in order to create a database for further investigations as well as to model the stability and control characteristics for the SACCON. This database comprises steady-state test results from various wind tunnels, including diverse flow regimes and aircraft configurations that accurately replicate the SACCON flight conditions. These conditions span both high and low speeds, along with multiple control surface deflections.

The unmanned aircraft under study has been the focus of various research efforts, see for example [13–17]. There one can find detailed information regarding its geometry, dimensions, flow characteristics, and extensive wind tunnel and numerical calculation results. In this article, we only showcase some of the most pertinent aircraft responses crucial for developing the S&C surrogate model. It is worthwhile to note that no hysteresis effects have been reported for the current dataset [15]. Figure 2 illustrates the characteristics of the SACCON pitching moment coefficient, C_m, at different Mach numbers as well as the influence of various control surface configurations. On the left side, nonlinearities in terms of the angle of attack are apparent. Additionally, we observe that different control surface settings have a minor impact on stability characteristics. On the right side, the effect of the angle of the sideslip is depicted. It is noteworthy that the sideslip angle shifts the pitch-up towards lower angles of attack, presenting a challenging scenario for modeling. The point

of reference for moments in this work is situated at 56.56% of the inner root chord length, measured from the apex. More details can be found in the references mentioned above.



(**a**) (**b**)

Figure 2. SACCON pitching moment coefficient characteristics. M and Re stand for Mach and Reynolds numbers respectively. Reynolds number in millions. OB: lower | upper flap deflection. (**a**) Control surfaces influence on C_m. (**b**) Sideslip angle influence on C_m. $M = 0.8, Re \approx 2.8$, baseline configuration.

3. Machine Learning Methodologies

In the previous section, it was stated that the outboard flap can actually perform a symmetrical split up, unlike the rest of the control surfaces. This capability must be transformed into numerical data in order to be suitable for the machine learning algorithms. In ML, encoding is crucial when dealing with non-numeric data, also known as categorical data, as it transforms these data into a format that preserves the information present in the original dataset. This allows the algorithm to understand the different categories and their relationships, which is essential for accurate modeling. For the encoding of the flap split, we have assessed two different techniques:

1. Upper and lower: we consider the upper and lower surfaces of the flap separately as two different independent input features.
2. Deflection and magnitude: we encode the split with two variables, one denoting the direction of the deflection, 1 for the same direction and -1 when opposite, and the second variable represents the deflection angle magnitude.

The ML training procedure, which will be discussed later in the paper, was conducted using both encoding techniques. The second encoding method produced superior results, so that is the approach adopted here. The criteria for determining superior results will be explained in Section 3.3.

Henceforth, the twelve features used to characterize the flow conditions and the aircraft configuration, which fully define our system and will further be used as an input for the surrogate model are as follows: angle of attack α, angle of sideslip β, Mach M and Reynolds Re numbers, along with the 6 control surfaces and the OB flap split encoding.

Additionally, it is well known that the Mach number dictates the flight regime of an aircraft and has a profound impact on its aerodynamic behavior, performance, and design considerations. Thus, we further analyze the SACCON behavior with respect to the Mach number in Figure 3. There we plot the pitching moment coefficient for different Mach number values. For ease of comparison, we keep the baseline configuration as well as similar Reynolds numbers across all polars. Besides, the sideslip angle can also be considered to be the same across all polars.

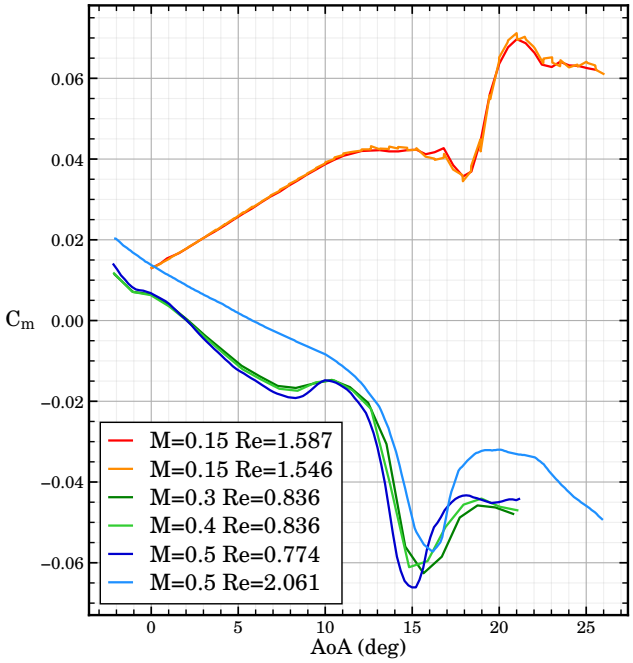


Figure 3. Flow regime effect on the pitching moment coefficient. Reynolds number in millions.

As expected, it can be observed that the aircraft's incompressible ($M < 0.3$) characteristics are completely different from the compressible ($0.3 \leq M$) behavior. Because of such significant differences, combining both flow regimes into a single surrogate model could potentially degrade the model's performance, specifically in terms of accuracy. Performing and assessing the results for both scenarios, a single and two separate surrogate models—one for each flow regime—confirmed our expectations; better results are achieved when developing two separate surrogate models, one for the incompressible and another one for the compressible flow regimes. Therefore, the latter is the approach that is adopted here. This same strategy was followed by Rajkumar et al. [18], although they set the regime boundary at Mach 0.4.

3.1. Data Preparation

The SACCON dataset consists of 40,374 static wind tunnel samples. Following standard machine learning practices, once the dataset has been properly preprocessed to address inconsistencies, duplicated entries, and potential issues such as invalid values, we divide it into an 80% portion for training, and a 20% portion designated for testing. A summary of the resulting datasets is given in Table 1 showing the number of AoA and AoS polars in each generated dataset as well as the total number of samples in each of them.

Subsequently, to extract meaningful insights and enhance our understanding of the data, we compute the correlation matrix as shown in Figure 4, and generate violin plots as depicted in Figures 5 and 6. The correlation matrix provides valuable information that can assist in making decisions related to model development and analysis. For example, it helps identify highly correlated features, which might be redundant, and features that have low or no correlation with the target variable, which might be less useful for prediction. Additionally, it allows us to explore linear relationships among the input features and

the target outputs, aiding in understanding which variables are likely to be important for the model. From there, we can see that there is a very low to null correlation between input features, which is favorable. However, we can also observe that the correlation between certain input features and the targets is also very small. This might indicate that these features might have very little to no influence on the output. The empty cells in the correlation plots for the corresponding flow regime are due to the variables being constant for that regime, as indicated by the violin plot. On the other hand, the violin plots provide a visual representation of the data distribution. We opt to plot the training and test datasets separately instead of combining them into one visualization because it enables us to further validate that the training dataset adequately represents the test dataset—which indeed is the case.

Table 1. Datasets overview.

	Num. Polars			
Dataset	α	β	**Num. Samples**	**Actual** %
Training (incomp.)	41	50	4142	10.26
Training (comp.)	182	25	28,391	70.32
Test (incomp.)	10	14	1037	2.57
Test (comp.)	54	8	6804	16.85

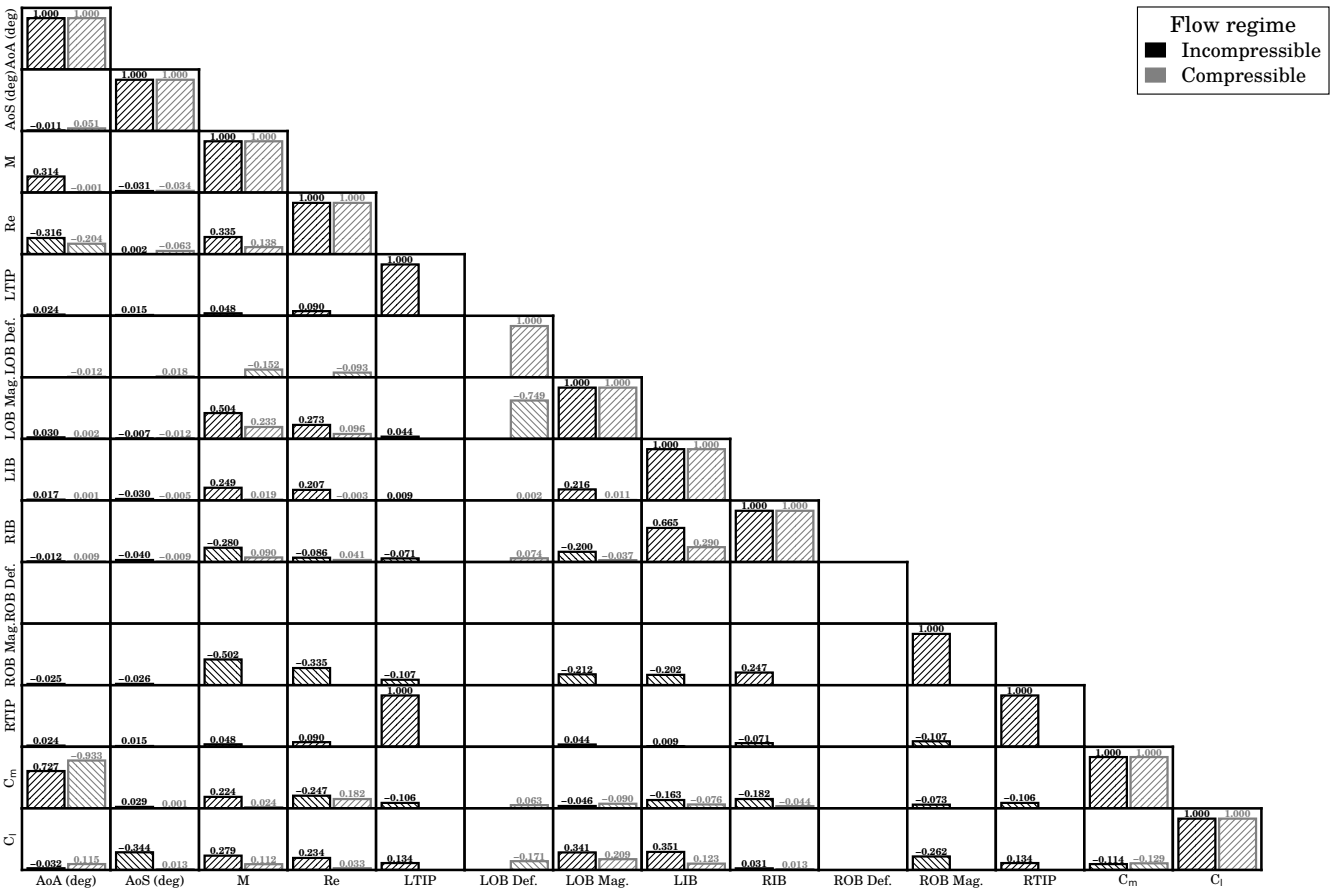
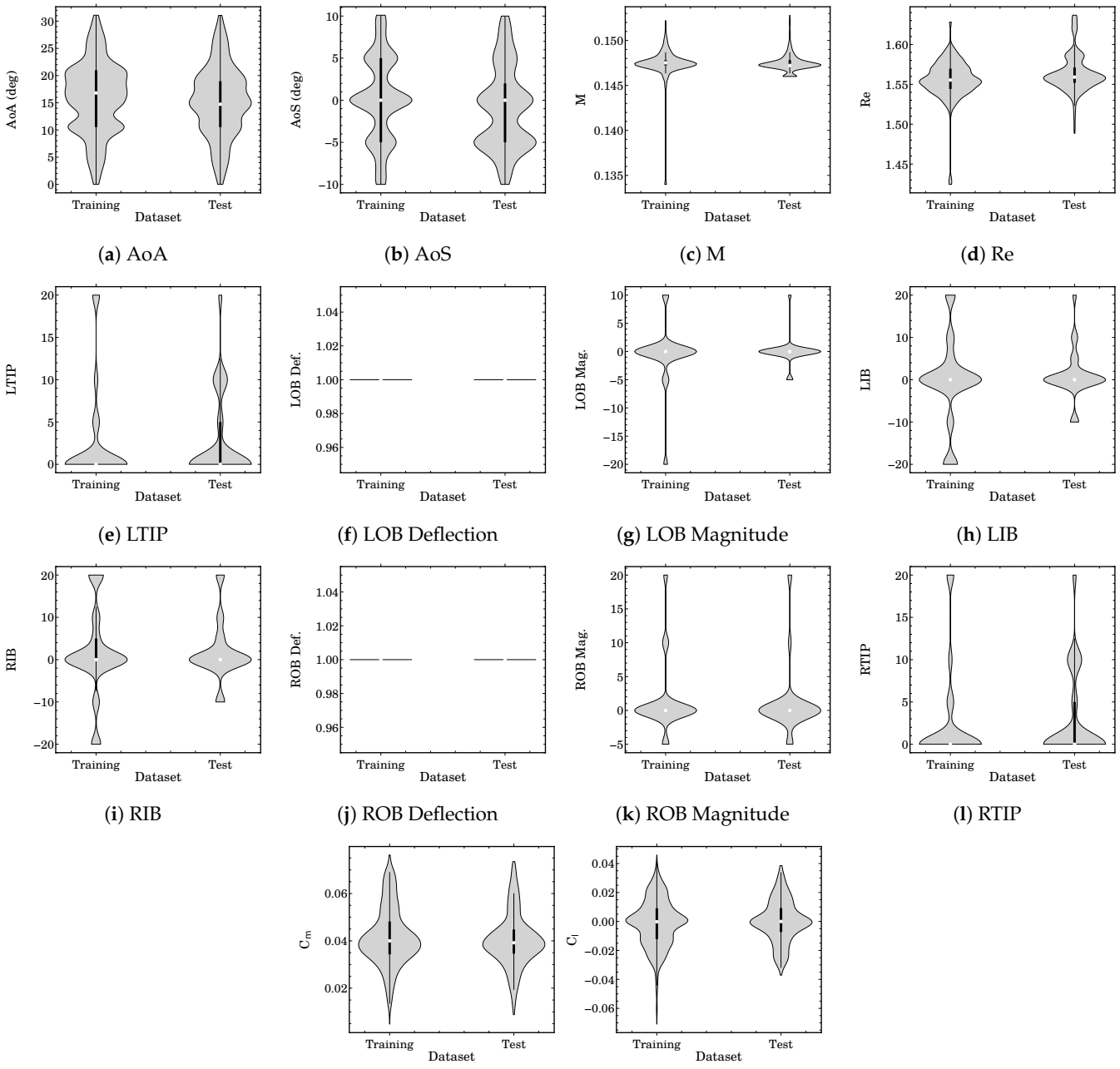


Figure 4. SACCON correlation matrix for both flow regimes.

One can observe that the correlation values of the roll moment coefficient with respect to the left and right control surfaces are not identical. This discrepancy may be attributed to the SACCON data. Specifically, asymmetric measurements of the deflections of the left and right control surfaces result in differing correlation values. This is evident in the distribution of the wind tunnel measurements shown in the violin plots.

The last step for the dataset preparation is to standardize the data, which is a crucial step for artificial neural networks. It helps to ensure that all features contribute equally to the model training process, speed up the training, and enhance stability and performance by mitigating issues related to numerical instability or exploding gradients. In this work, we use the so-called Z-score standardization technique. The mean and standard deviation from the training dataset is used to normalize both datasets, training and test. These values are reported in Table 2 alongside the minimum and maximum values for each input feature. The data standardization step is only applied to ANN as the ensemble learning models do not require it, as it could even deteriorate their performance.

Finally, we note that variables that are constant for the given flow regime have been excluded as input features for the ML models, as they do not provide any additional information.



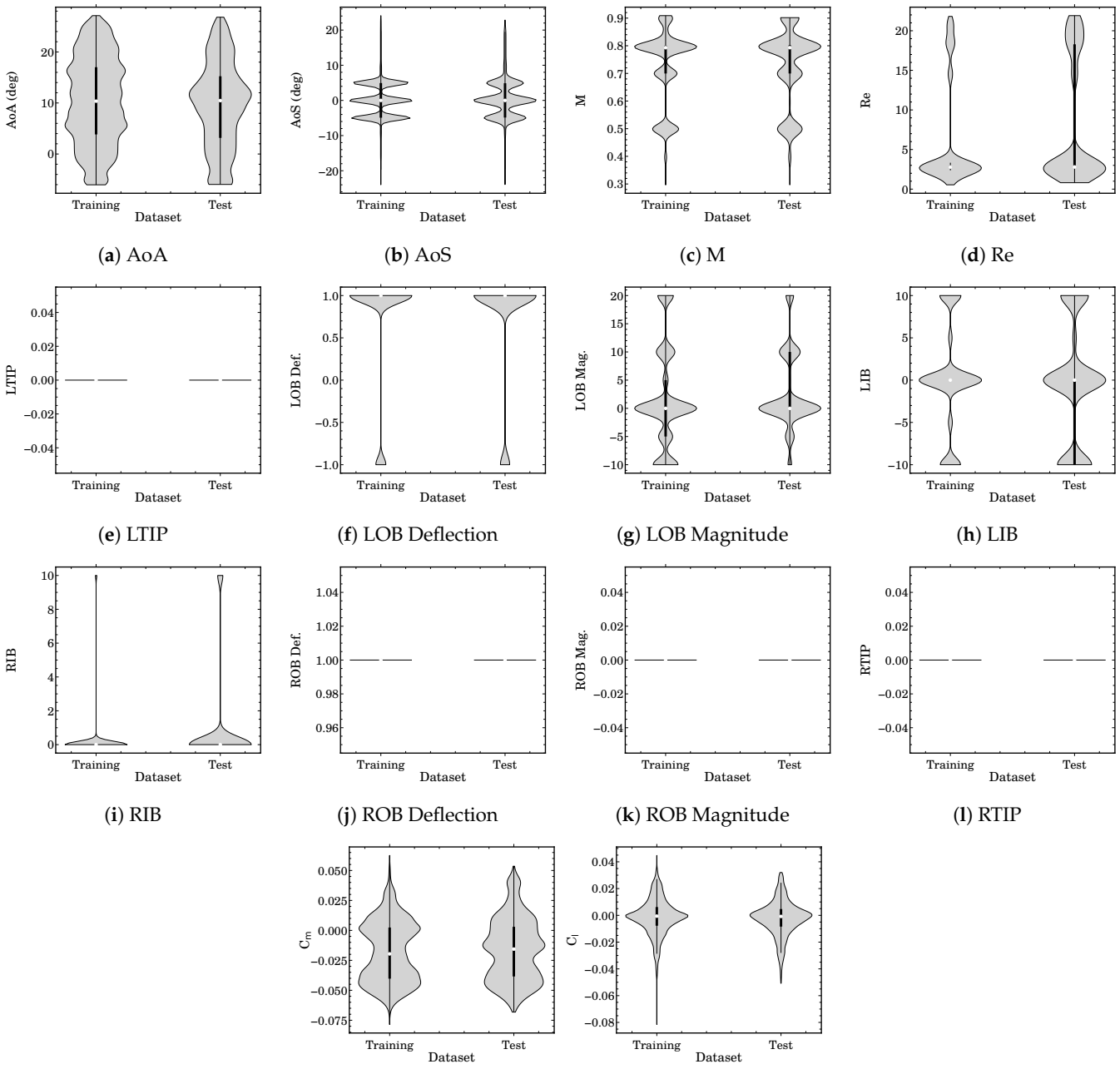
(**a**) AoA (**b**) AoS (**c**) M (**d**) Re

(**e**) LTIP (**f**) LOB Deflection (**g**) LOB Magnitude (**h**) LIB

(**i**) RIB (**j**) ROB Deflection (**k**) ROB Magnitude (**l**) RTIP

(**m**) C_m (**n**) C_l

Figure 5. Incompressible violin plots.



(**a**) AoA (**b**) AoS (**c**) M (**d**) Re

(**e**) LTIP (**f**) LOB Deflection (**g**) LOB Magnitude (**h**) LIB

(**i**) RIB (**j**) ROB Deflection (**k**) ROB Magnitude (**l**) RTIP

(**m**) C_m (**n**) C_l

Figure 6. Compressible violin plots.

Table 2. Training dataset input features range, mean and standard deviation values. *Re* in millions.

	AoA	AoS	M	Re	LTIP	LOB.D	LOB.M	LIB	RIB	ROB.D	ROB.M	RTIP
(**a**) Incompressible.												
Min	0	−10	0.134	1.425	0	1	−20	−20	−20	1	−5	0
Max	31.09	10.1	0.152	1.628	20	1	10	20	20	1	20	20
Mean	15.65	9.2×10^{-2}	0.147	1.555	2.563	1	−0.57	−0.469	1.391	1	1.285	2.56
Std	6.531	5	2×10^{-3}	2.4×10^{-2}	5.84	0	5.3	10.16	10.08	0	5.173	5.84
(**b**) Compressible.												
Min	−6.035	−24	0.3	0.545	0	−1	−10	−10	0	1	0	0
Max	27.12	24.11	0.909	21.81	0	1	20	10	10	1	0	0
Mean	10.386	0.18	0.726	6.248	0	0.723	1.447	0.058	0.227	1	0	0
Std	8.468	5.364	1.36	6.782	0	0.691	8.757	6.234	1.489	0	0	0

3.2. ML Models

Artificial neural networks are well-known for their ability to efficiently process large datasets. When trained with an adequate amount of data, they excel at capturing nonlinearities within physical models. This enables the development of real-time applications based on the knowledge they acquire. They can be optimized using diverse data sources, combining numerical simulation results, wind tunnel, and/or flight measurements as training data. The robustness, error tolerance, generalization ability, and performance of ANNs align precisely with the characteristics of aerodynamic surrogate model goals, making them a promising solution for aerodynamic calculations.

However, the assessment of the nature and quality of the SACCON data revealed that the wind tunnel data might be not very favorable for ANNs due to insufficiently informative features. This limitation may hinder ANNs from recognizing underlying global problem trends, resulting in poor generalization. Conversely, ensemble learning techniques can sometimes surpass neural networks in handling tabular data [19], owing to several critical advantages they offer. For instance, EL offers diverse modeling approaches that enable the capture of various patterns and relationships present in tabular data. Therefore, ensemble learning models such as Random Forests (RF) [20], Extreme Gradient Boosting (XG Boosting) [21], and AdaBoost (Adaptive Boosting) [22] have been explored, as they might be better suited for the given problem.

Nevertheless, in our investigation, it was observed that XG Boosting exhibited superior performance compared to other ensemble learning techniques. Consequently, subsequent sections of this paper will exclusively present findings pertinent to XG Boosting in the context of EL results.

During the training or learning phase of the ANN, the optimizer utilizes a cost or loss function to iteratively update the model parameters. Such a function must measure the discrepancy between the ground truth values, Y_i, and the model prediction, $\widehat{Y}_i$. The objective is to minimize the loss function, thereby improving the model's predictive accuracy. As commonly conducted for regression problems, we use the L_2-regularized mean squared error,

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2 + \lambda \sum_{j=1}^{m}\theta_j^2. \tag{1}$$

Here θ denotes the model parameters, n is the total number of samples in the mini-batch, and m is the total number of weights in the ANN. The second summation in the previous equation denotes the regularization term whose strength is controlled by the weighting factor λ. In essence, it adds a regularization term to the loss function during training, which penalizes large weights in the network and hence prevents overfitting. As for the optimizer, in this work, we use the Adaptive Moment Estimation (Adam) [23]. The benefits of Adam over other traditional optimizers, such as stochastic gradient descent, include faster convergence, better handling of noisy or sparse gradients, and reduced sensitivity to hyperparameters. Here we use Adam with a constant learning rate, the value of which is provided in Section 4. Besides, the optimizer has two hyperparameters that control the exponential decay rates for the first and second moments estimated from the gradients, commonly known as β_1 and β_2, respectively. As in typical practice, we do not fine-tune these two parameters; instead, we use the recommended values $\beta_1 = 0.9$, $\beta_2 = 0.999$ [23].

Analogously XG Boosting is also trained by minimizing the loss of an objective function [21]. Here we also adopted an L_2-regularized mean squared error.

Revisiting the SACCON data, we can notice from the correlation plots the low inference of certain input features. It is a good practice to have correlation values predominately above 0.5. Hence, we have also explored the application of data dimensionality reduction techniques to create a more informative, albeit lower-dimensional space composed of combinations of these input features. We looked at three different methods, each one being explained in its own section below.

3.2.1. Autoencoders

Autoencoders consist of an encoder and a decoder neural network, which work in tandem: the encoder compresses the input data into a lower-dimensional representation, while the decoder aims to reconstruct the original input from this compressed representation. During training, the autoencoder learns to minimize the reconstruction error between the input and the output. Once trained, the encoder can be utilized independently to extract meaningful features or reduce the dimensionality of new data. This makes autoencoders valuable in scenarios where data are scarce or when capturing intrinsic data patterns is crucial.

3.2.2. Principal Component Analysis

Principal component analysis (PCA) [24] is a dimensionality reduction technique by which a reduced base is computed. By projecting the original data onto this reduced-dimensional space defined by the selected principal components, dimensionality reduction can be achieved.

3.2.3. Partial Least Squares Regression

Partial Least Squares (PLS) [25] works by finding linear combinations of the original predictor or input variables that explain both the variance in the predictors and their covariance with the response or target variable. Analogously to PCA, the original data can be reduced by projecting onto the latent variables, known as components.

Then, the ML training procedure was also repeated using the previous dimensionality reduction techniques on the SACCON dataset. The obtained results as well as the training time were nearly identical to what will be presented later, indicating that no benefit was gained from using dimensionality reduction. Therefore, the input space for the ML models is the same one explained in Section 3.

The implementation has been completely written in Python. The ANN modules have been carried out with PyTorch [26] while the XG Boosting code was developed using the XGBoost [21] framework. All training has been conducted using central processing units (CPU).

3.3. Hyperparameter Tuning

Each machine-learning model relies on specific hyperparameters, which are predetermined settings that govern the behavior and complexity of the model. The optimization of these hyperparameters is crucial, as it has a direct impact on the performance, predictive accuracy and generalization capabilities of the models. Hyperparameter tuning involves selecting the most effective combination of hyperparameter. For instance, in this study, we employ a k-fold cross-validation technique [27] for the hyperparameter optimization. We refer to the previous reference for a detailed explanation of this methodology. One of the main and crucial aspects of employing k-fold cross-validation is the careful definition of the loss function or score. It serves as a metric for evaluating the performance of the machine learning model across different hyperparameter settings. Moreover, different metrics emphasize different aspects of model performance. Given the nature of the problem, it is essential to consider not only the accuracy of the predictions but also their compliance with specified tolerance intervals around the target values. Therefore, the loss function has been defined as

$$L = \text{RMSE} \times (1 + |k_1 \log(r_C) + k_2 \log(r_{dC})|), \tag{2}$$

where RMSE is the Root Mean Square Error of the target variable(s) providing a measure of the model's accuracy by quantifying the average magnitude of prediction errors, r_C represents the ratio of points falling within the tolerance margin for the target variable(s), r_{dC} denotes the ratio of points within the tolerance margin for the derivative of the target variable(s) with respect to the polar angle, either the angle of attack or the sideslip angle, and k_1 and k_2 are weighting constants used to control the relevance of each factor. For this work, we set both of them to one. Therefore, the k-fold cross-validation procedure will

identify the outperforming parameter combination that achieves the highest accuracy while ensuring the maximum number of points fall within the specified tolerances.

For the cross-validation, the training dataset must be divided into a suitable number of groups or folds. This choice significantly influences the model generalization performance. While a larger number of folds is generally favored for statistical robustness, it is crucial to find a balance between the available sample size and computational resources. In our study, we determined that using five folds effectively balanced these considerations. In addition, we repeat the whole k-fold cross-validation process twice. In Table 3 we show the hyperparameter space for the ANN and the EL models, respectively.

Table 3. ML models hyperparameters space.

Parameters	**Range of Values**
(**a**) Artificial neural network.	
Epochs	250, 500
Mini batch size	64, 128
Hidden layers	3, 4, 5, 6
Neurons per layer	50, 150, 250
Activation func.	`ReLU`, `Tanh`, `Sigmoid`
Dropout	`No`, 0.5, 0.75
Batch normalization	`Yes`, `No`
Regularization	$[0.001, 0.01, 0.1, 1] \times 10^{-1}$
Opt. learning rate	$1 \times 10^{-4}, 1 \times 10^{-3}$
(**b**) XG Boosting. We refer to [21] for a detailed definition of each parameter.	
Num. estimators	$[0.01, 0.1, 0.5, 1, 1.5, 2, 3, 5] \times 10^{3}$
Max tree depth	`None`, 5, 10, 25
Num. features	5, 10, 12
Max. num. leaves	`None`, 16, 64
Learning rate	$[0.1, 1, 3] \times 10^{-1}$
Regularization	$[0.01, 0.1, 1] \times 10^{-1}$
Subsample	0, 0.1, 0.4, 0.7, 1
Gamma	$0, 1 \times 10^{-2}, 1$

Regarding the ANN architecture, the model is composed of fully connected layers. The actual number of neurons per layer is determined by generating combinations using the values in `Neurons per layer`. These combinations can vary based on the number of layers in the ANN.

Before delving into the results, we first want to explain how the machine learning models handle the data for training and validation. Initially, from the wind tunnel measurements, we compiled a collection of polars, each comprising a specific quantity of measurements, termed as samples herein. During the training phase, when the primary objective is to forecast the target variable(s), each sample is randomly selected and treated in isolation, devoid of any interdependence with the others. However, during evaluation, it becomes imperative to compute the derivative of the target variable(s) concerning the polar angle, see Equation (2). This necessitates the preservation of the polar information and its sequential arrangement preserving the integrity of the polar. The derivatives are computed using central finite differences except for the extremes, where forward or backward methods are employed instead. Finally, once the machine learning model has been trained, the model will produce single predictions at a time—the model predicts the target variable(s) for a given a set of input features.

3.4. Multiple Outputs Models

Single-target machine learning techniques including both, artificial neural networks and ensemble learning algorithms, are straightforward to define and train. However, the landscape becomes more intricate when confronted with multiple targets. For instance, straightforward ensemble learning methods cannot directly handle multiple outputs simultaneously. Some research has been conducted into addressing this issue, see for example

the work from [28], yet significant challenges persist. In contrast, artificial neural networks circumvent this challenge, given their inherent capacity to accommodate multiple outputs. Nevertheless, a more foundational question emerges: does training a model with multiple outputs offer advantages over employing a separate model for each target variable?

On one hand, the Single Model with Multiple Outputs (SMMO) strategy, often referred to as a joint model, involves the creation of a single machine learning algorithm capable of producing multiple outputs, such as pitching and rolling moment coefficients. On the other hand, the Multiple Models with Single Output (MMSO) strategy, also termed a disjoint model, relies on developing separate machine learning models for each variable of interest. The SMMO strategy typically leads to lower model capacity per output when compared to the disjoint model. However, as long as computational resources are available, increasing the capacity is usually a straightforward task. When prediction tasks exhibit interrelatedness, implying correlation or covariance between output values, training a unified multi-output model can offer potential advantages, resulting in improved predictive performance. This phenomenon represents a form of knowledge transfer, with the algorithm learning valuable hidden representations applicable to predicting all outputs [29–31]. Consequently, these representations embody knowledge that benefits all the data. Typically, the SMMO approach may entail a more time-intensive training process in comparison to the MMSO strategy. However, in general, determining which approach is superior often necessitates practical experimentation with both methodologies. In the next section we will investigate and present the results for both approaches.

4. Results

At first, we assess the results for the surrogate model developed considering the pitching moment coefficient as the only target variable. These are discussed in Section 4.1. Then in Section 4.2 we showcase the results when multiple targets are considered.

4.1. Pitching Moment Coefficient

We start by presenting the cross-validation results. First, the outperforming hyperparameters for each ML technique are presented in Tables 4 and 5. The cross-validation scores corresponding to the best hyperparameter combination are reported in Table 6. Note, that these values report the combined or joint model scores. Recall that an incompressible and a compressible model are trained individually. Then, we combine both sets of results into a single joint model which is further used for comparison.

Table 4. Extreme Gradient Boosting outperforming parameters.

	Value	
Parameter	**Incompressible**	**Compressible**
Num. estimators	1500	1000
Max tree depth	`None`	10
Num. features	10	8
Max. num. leaves	`None`	64
Learning rate	0.1	0.3
Regularization	0.01	0.01
Subsample	0.4	0.4
Gamma	0	0

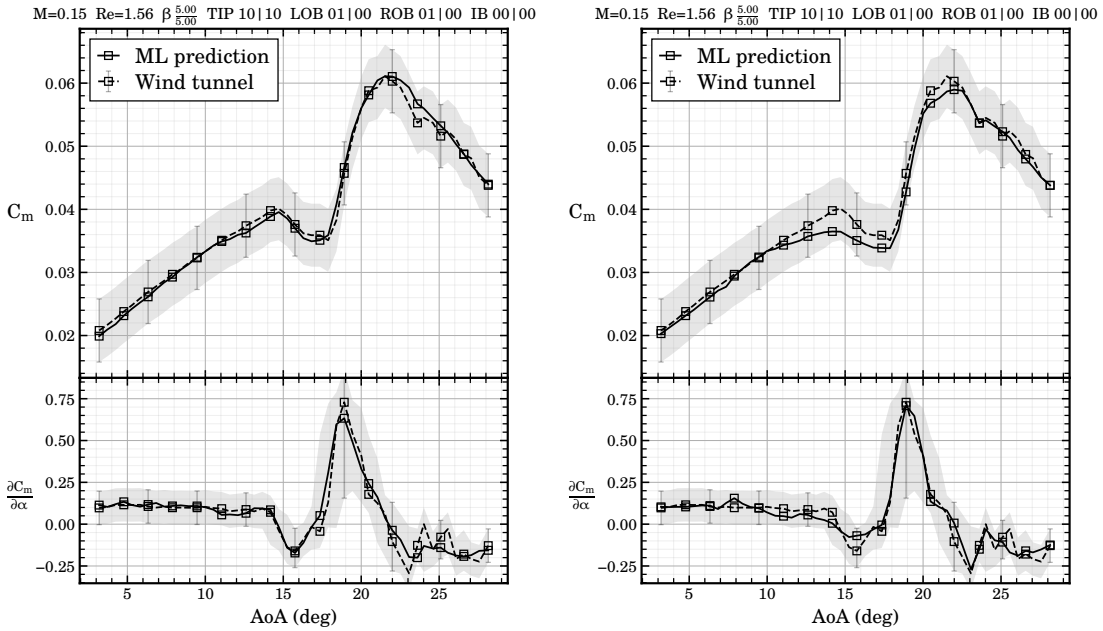
Table 5. Neural network outperforming parameters.

	Value	
Parameter	**Incompressible**	**Compressible**
Epochs	500	500
Mini batch size	64	64
Hidden layers	4	5
Neurons per layer	250, 150 × 2, 50	250 × 3, 50, 150
Activation function	ReLU	ReLU
Dropout	No	No
Batch normalization	No	No
Regularization	1×10^{-2}	1×10^{-1}
Opt. learning rate	1×10^{-3}	1×10^{-3}

Table 6. ML models joint k-fold cross validation score. The lower the better.

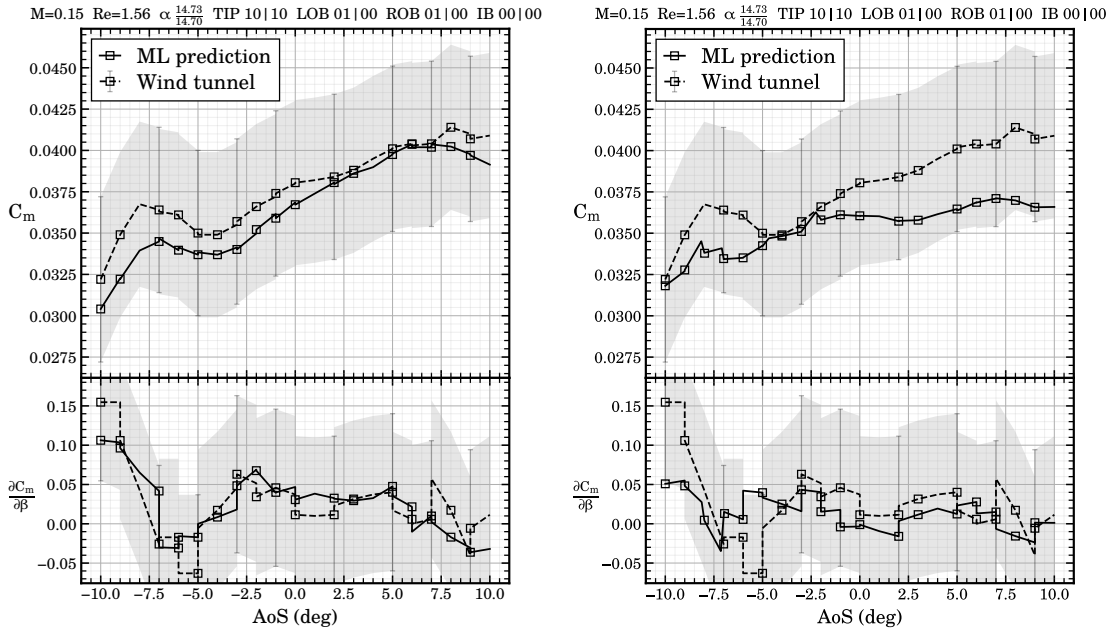
Model	**Score**
Neural Network	2.421×10^{-3}
XG Boosting	3.104×10^{-3}

Both methodologies exhibit similar cross-validation scores; however, the ANN surpasses XG Boositng by achieving a lower score, thus demonstrating to be better aligned with our specific requirements and objectives. Additionally, Figures 7 and 8 provide a comparison between the predicted and target C_m values for a test polar, wherein the flow conditions and aircraft configuration are summarized atop each polar. Specifically, from left to right, the Mach number, Reynolds number (millions), range of angle of attack or sideslip angle values (with low and high bounds indicated at the bottom and top, respectively), TIP deflection for the left | right wings, left-wing OB deflection direction and magnitude, analogously the right-wing OB, and finally the IB deflection for the left | right wings. These figures encompass data from a polar not utilized during the training process. The gray area denotes the tolerances margin.



(**a**) Neural Network (**b**) XG Boosting

Figure 7. Machine learning results for a test AoA polar.



(**a**) Neural Network (**b**) XG Boosting

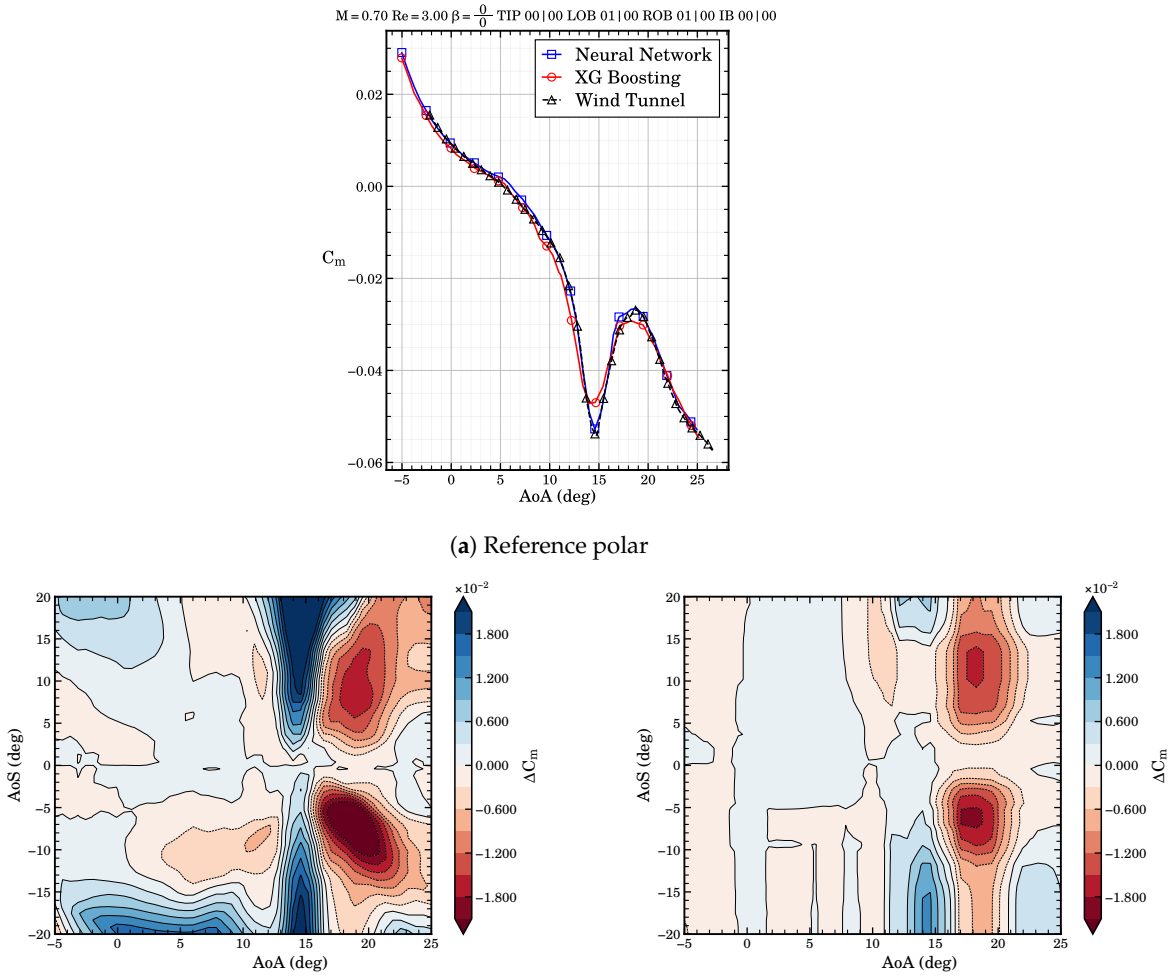
Figure 8. Machine learning results for a test AoS polar.

The agreement between both methods and the experimental measurements is excellent. Both methods completely catch the pitch up at around AoA $= 19°$ in Figure 7. Besides, it can also be seen that the derivative of the pitching moment coefficient with respect to the polar angle is also inside the tolerances. Comparatively, the ANN response for C_m exhibits smoother trends across both the AoA and the AoS polars, in contrast to the less smooth response observed with XG Boosting for the sideslip angle polar. A stepping response can be observed in Figure 8b around AoS $= -8°$.

Finally, we generate a heat map plot to visualize the response of both ML models when two input features are changed while the rest of the parameters are kept constant. This technique helps to diagnose the models, for example identifying potential overfitting. In this case, we create the response surface when the angle of attack and the sideslip angle are changed, see Figure 9. The AoA ranges from -5 to 25 degrees while the AoS from -20 to 20 degrees. We have first established the baseline values of C_m under specific conditions: zero angle of sideslip and the whole range of angle of attack. These baseline values serve as our reference points. Then instead of directly depicting the C_m values in the heat map, we illustrate the deviation of the pitching moment coefficient with respect to the corresponding (same value of angle of attack) baseline value, ΔC_m, aiding in the interpretation of how C_m deviates under different circumstances.

This approach not only aids in understanding the nuanced behavior of C_m under different conditions but also facilitates the identification of key factors driving its variation. For example in terms of deviation values, we can better discern patterns and trends, ultimately enriching our comprehension of the ML models.

The assessment seemingly revealed some problems in terms of predictive or extrapolation capabilities for both models. For instance, in a symmetric aircraft configuration, one would anticipate a symmetric response for the angle of the sideslip. However, as observed, neither method predicts a perfectly symmetrical response. The first impression could be that this discrepancy is attributed to a poorly trained model. Yet, as depicted in Figure 8, the wind tunnel measurements themselves are not entirely symmetrical. Nonetheless, the ensemble learning model demonstrates an evident independent prediction concerning the sideslip angle for low and negative angles of attack, not properly capturing the nonlinearities of the model.



(**a**) Reference polar

(**b**) Neural Network

(**c**) XG Boosting

Figure 9. ML models results comparison. (**a**) Polar results used as a reference. (**b**,**c**) Heat maps for $\alpha - \beta$. $M = 0.7, Re = 3 \times 10^6$. Aircraft configuration is the baseline. The heatmaps show the variation in C_m with respect to the reference polar.

In contrast, the neural network response seems to be smoother and physically accurate to what would be expected. The inconsistencies observed in the ensemble learning heat map might be associated with the fundamental mechanism of the method, i.e., decision trees. Decision trees optimize threshold boundaries which are used to create nodes and leaves, later used to split and classify the data in order to predict the output. The investigated space in Figure 9 is mostly unseen by the machine learning algorithms during training. Because of that, the classification might lead to incorrect values. In contrast, neural networks use regression functions to map the data to the output; therefore, they tend to capture the nonlinearities behind the physical model and ultimately make better extrapolations.

We can then conclude that neural networks possess better predictive or generalization capabilities, and therefore, constitute a more consistent architecture for the SACCON surrogate model than XG Boosting. Previous observations alongside the discussion presented in Section 3.4 led to the decision to discard ensemble learning techniques as surrogate models. In line with our findings, Volpiani [32] reached similar conclusions, finding that ANNs provide a smoother response and better generalization performance compared to RF when augmenting Reynolds-averaged Navier–Stokes simulations. Although these conclusions were drawn in a different research context, they align precisely with our findings regarding the efficiency of ANN in interpolating and extrapolating output quantities. Henceforth, in the next section only the results for the artificial neural networks will be shown.

4.2. Pitching and Rolling Moment Coefficients

In this section, we are interested in predicting more than one aerodynamic property. In this regard, we have selected the pitching, C_m, and rolling, C_l, moment coefficients. As discussed in Section 3.4 we investigate two different approaches, the SMMO and the MMSO. Besides, as stated in the previous section, only the results for the artificial neural networks will be reported.

4.2.1. MMSO

For this approach we train two models independently, one for each variable. Here we use the term *model* to refer to the combination of two surrogate models. Recall that we train an incompressible and a compressible surrogate. The outperforming hyperparameters are reported in Tables 5 and 7. The k-fold cross-validation metrics are summarized in Tables 8–10. There, the R^2 and p-value from the linear regression between the ground truth and predicted values are documented, along with the RMSE. While the former two metrics are not utilized for the k-fold score calculation in Equation (2), they contribute to additionally assessing the results. It is important to note that the presented values represent the averages across all k-fold evaluations.

Table 7. C_l-MMSO ANN outperforming parameters.

	Value	
Parameter	**Incompressible**	**Compressible**
Epochs	500	500
Mini batch size	64	64
Hidden layers	6	6
Neurons per layer	250 × 2, 50, 150 × 3	250 × 3, 150, 250 × 2
Activation function	ReLU	ReLU
Dropout	`No`	`No`
Batch normalization	`No`	`No`
Regularization	1×10^{-1}	1×10^{-2}
Opt. learning rate	1×10^{-3}	1×10^{-3}

Table 8. MMSO metrics.

	C_m			C_l		
Model	R^2	p-**Value**	**RMSE**	R^2	p-**Value**	**RMSE**
Incompressible	0.984	0.0	1.817×10^{-3}	0.9122	0.0	4.31×10^{-3}
Compressible	0.993	0.0	2.312×10^{-3}	0.932	0.0	3.532×10^{-3}
Joint	-	-	2.26×10^{-3}	-	-	3.672×10^{-3}

While the reported metrics indicate that the surrogate model also achieves high precision for C_l, its accuracy for C_l is comparatively lower than its accuracy for C_m. These findings are also confirmed by the linear regression results. While the C_m models show R^2 values around 0.99, meaning almost a perfect match, the regression values for the C_l model suggest a worse fit for the training data. The number of points inside tolerances for C_l has also been reduced in comparison with C_m. This results in a worse performance as per our evaluation criteria, as seen in Table 10.

Table 9. MMSO number of entries inside | outside the tolerances.

Model	C_m		$\partial C_m/\partial\alpha$		$\partial C_m/\partial\beta$	
(**a**) C_m						
Incompressible	811.4	19.1	448.2	17.5	355.3	11.4
Compressible	5417.1	269.3	4754.6	219.8	642.3	520.4
Joint	6228.5	288.4	5202.8	237.3	997.6	531.8
Model	C_l		$\partial C_l/\partial\alpha$		$\partial C_l/\partial\beta$	
(**b**) C_l						
Incompressible	538.5	288.2	339.1	120.6	240	125.6
Compressible	3987.2	1688.4	3610.2	1375.5	341.9	342.7
Joint	4525.7	1976.6	3949.3	1496.1	581.9	468.3

Table 10. MMSO cross validation scores.

Model	C_m	C_l
Incompressible	1.864×10^{-3}	5.771×10^{-3}
Compressible	2.389×10^{-3}	4.59×10^{-3}
Joint	2.421×10^{-3}	4.811×10^{-3}

Comparisons between the number of layers and neurons per layer for the pitching and rolling moment coefficients, as shown in Tables 5 and 7, indicate an increase in the model's complexity for C_l. One potential explanation for this is that the wind tunnel data for C_l appear to be more prone to noise. For instance, fluctuations are noticeable in Figure 10 for angles of sideslip within the range $[-20, -15]$ degrees. One can note that the regularization factor used for the incompressible C_l-MMSO is the largest, indicating that the ANN aims to smooth out some of the noise.

4.2.2. SMMO

Now, we consider both target variables simultaneously and thus train a single model with multiple outputs. The outperforming hyperparameters for each neural network are reported in Table 11, while the k-fold cross-validation metrics are summarized in Tables 12–14.

Table 11. SMMO ANN outperforming parameters.

	Value	
Parameter	**Incompressible**	**Compressible**
Epochs	500	500
Mini batch size	64	64
Hidden layers	6	5
Neurons per layer	250 × 2, 150, 50, 250, 150	250, 50, 150 × 2, 50
Activation function	ReLU	ReLU
Dropout	`No`	`No`
Batch normalization	`No`	`No`
Regularization	1×10^{-2}	1×10^{-2}
Opt. learning rate	1×10^{-3}	1×10^{-4}

Table 12. SMMO RMSE for C_m and C_l.

Model	C_l	C_m	$C_l + C_m$
Incompressible	3.34×10^{-3}	2.04×10^{-3}	3.899×10^{-3}
Compressible	3.059×10^{-3}	2.255×10^{-3}	3.817×10^{-3}
Joint	3.09×10^{-3}	2.228×10^{-3}	3.836×10^{-3}

Table 13. SMMO number of entries inside | outside the tolerances for C_m and C_l.

Model		Coeff.		$\frac{\text{Coeff.}}{\partial\alpha}$		$\frac{\text{Coeff.}}{\partial\beta}$	
C_l	Incompressible	320.3	103.3	162.7	67.5	121.4	61.1
	Compressible	2085	754.2	1836.6	657.3	184.7	161.5
	Joint	2405.3	857.5	1999.3	724.8	306.1	222.6
C_m	Incompressible	402.2	12.3	228.8	9.2	175.2	5.3
	Compressible	2717.7	125.2	2345.6	147.3	326.1	21.5
	Joint	3119.9	137.5	2574.4	156.5	501.3	26.8
$C_l + C_m$	Incompressible	717.5	114.3	381.7	74.2	298.8	68.6
	Compressible	4798.2	875.2	4174.8	804.3	509	184.1
	Joint	5515.7	989.5	4556.5	878.5	807.8	252.7

Table 14. SMMO cross validation scores.

Model	C_l	C_m	$C_l + C_m$
Incompressible	4.322×10^{-3}	2.16×10^{-3}	4.477×10^{-3}
Compressible	3.89×10^{-3}	2.313×10^{-3}	4.409×10^{-3}
Joint	3.911×10^{-3}	2.293×10^{-3}	4.468×10^{-3}

The resulting model presents a more complex architecture than the standalone C_m-MMSO. The exact reason for this is unclear. It could be due to the complexity introduced by predicting multiple outputs, or as discussed in Section 4.2.2, it might stem from the inherent complexity associated with predicting C_l.

By comparing Tables 8 and 12 it is evident that the accuracy for the rolling moment coefficient under the SMMO approach has been increased compared to the MMSO, while the accuracy for the pitching moment coefficient remained nearly the same, with a slightly increase. In order to facilitate the comparison between both approaches, we gather the k-fold scores into a single table, Table 15. We can, therefore, conclude that, for the SACCON context, the SMMO approach yielded superior results, displaying a lower k-fold score and hence demonstrating a better performance according to the established criteria, which considers both prediction accuracy and tolerance compliance. Additionally, in Figures 10 and 11 we compare the same test polars computed with the SMMO and the MMSO strategies. Once again, the agreement between the ANN model and the wind tunnel measurements is more than satisfactory. Still, the surrogate model does not fully capture the behavior of both moment coefficients perfectly. For C_m the largest discrepancy is observed in the sideslip polar, see Figure 10c, around AoS $= -24°$, which is at the extreme end of the AoS training range and thus expected to have a larger error. Specifically, for the roll moment coefficient, the ANN is not capable of predicting the sudden change in characteristics at AoS $= -10°$. Although differences are noticeable, the accuracy of the neural networks using the SMMO approach surpasses that of the MMSO, thereby minimizing these discrepancies. For example, the pitch-up behavior at an angle of attack AoA $= 9°$ in Figure 11 is not well captured by the MMSO strategy, whereas the SMMO does capture it. These findings suggest that, for the SACCON dataset and our objectives, the SMMO strategy is more suitable.

Table 15. ANN cross validation score. The lower the better.

	C_l-MMSO	C_m-MMSO	$C_l + C_m$-SMMO
C_l	4.811×10^{-3}	–	3.911×10^{-3}
C_m	–	2.421×10^{-3}	2.293×10^{-3}

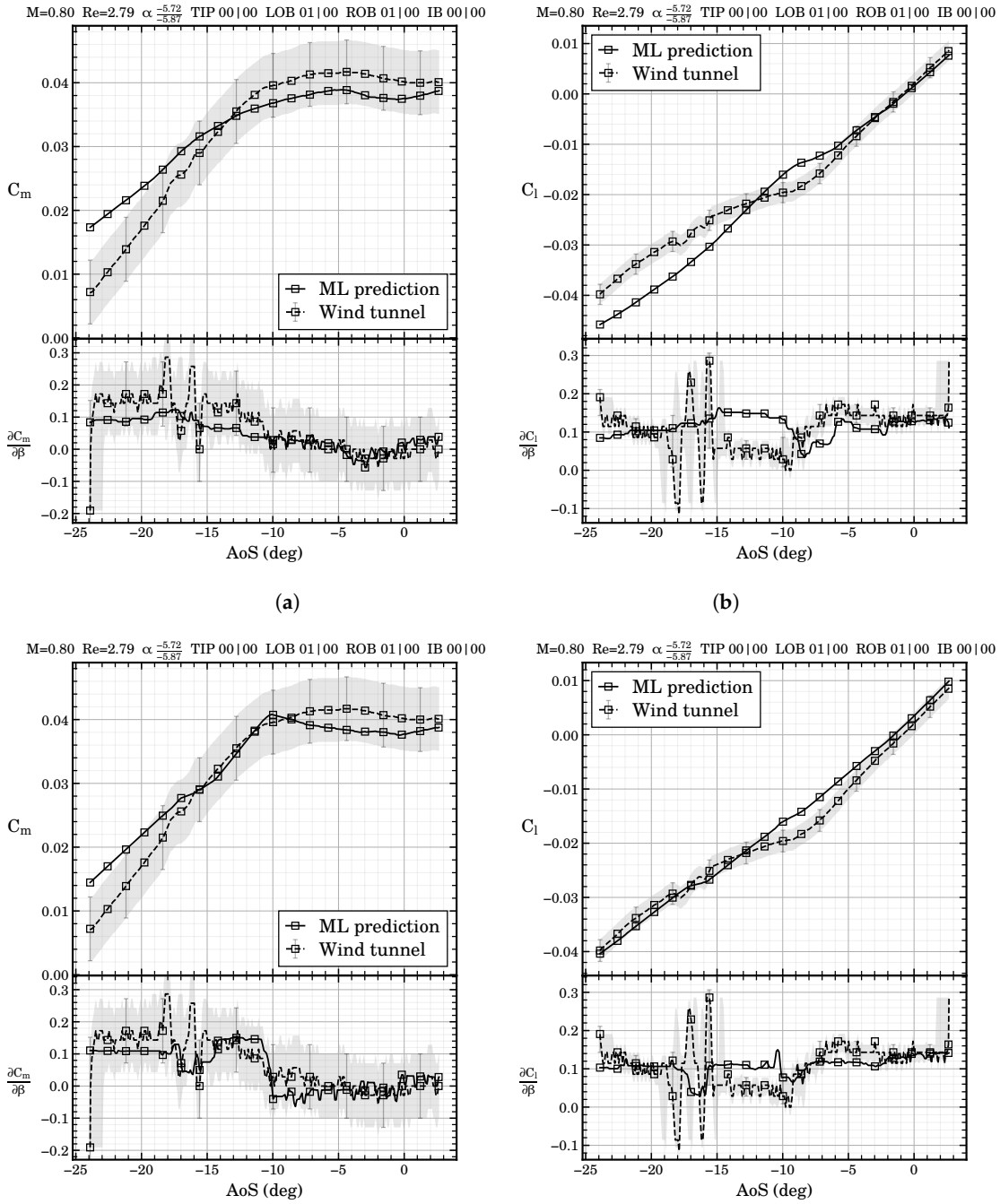


Figure 10. ANN results for a test AoS polar: (**a**) Trained using C_m-MMSO. (**b**) Trained using C_l-MMSO. (**c**,**d**) Trained using C_l + C_m-SMMO.

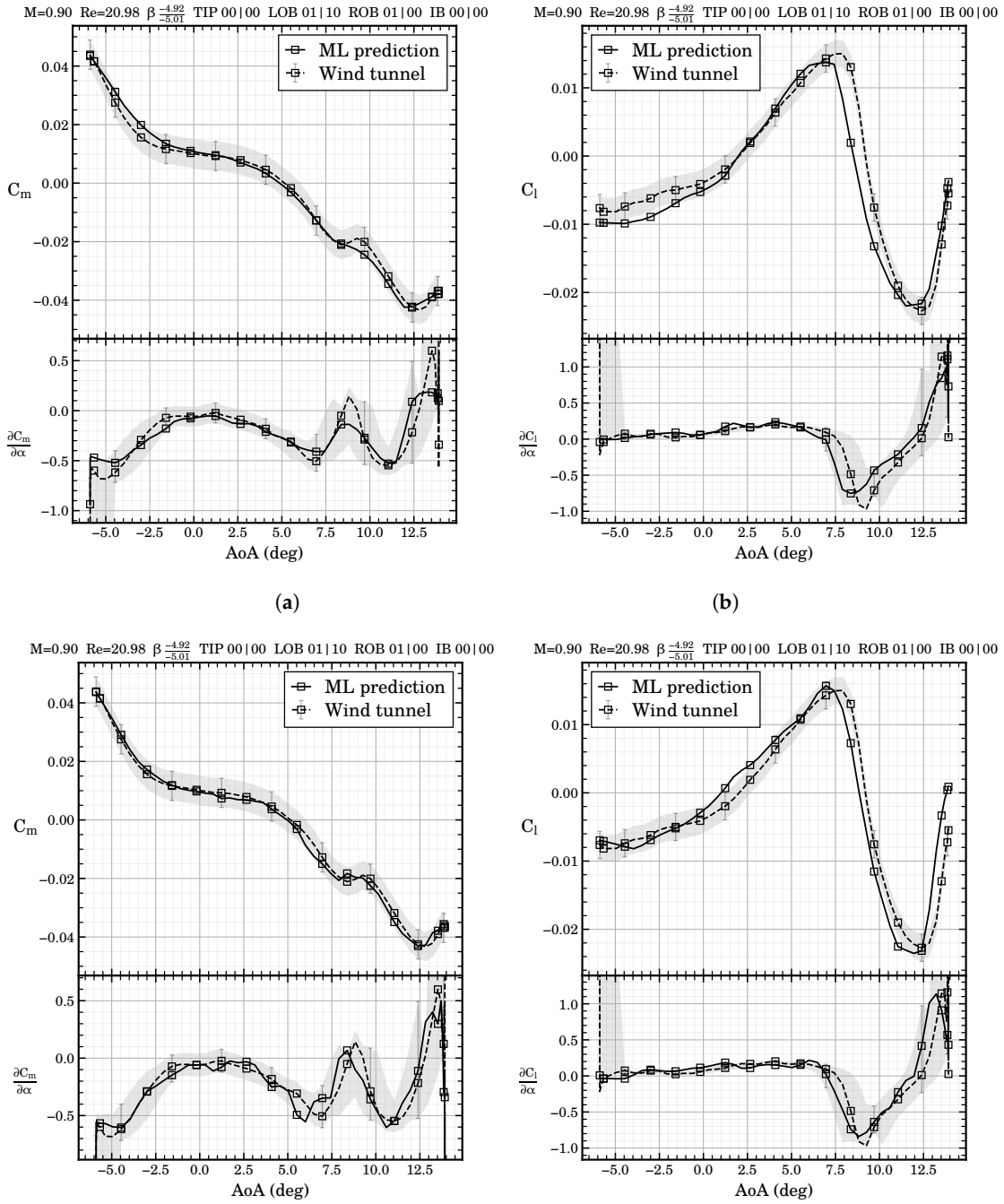


Figure 11. ANN results for a test AoA polar: (**a**) Trained using C_m-MMSO. (**b**) Trained using C_l-MMSO. (**c**,**d**) Trained using C_l + C_m-SMMO.

Finally, Table 16 compares the root mean squared error for the test dataset computed using both strategies. The error calculated for entirely unseen data confirms that the network exhibits greater precision for the pitching moment than for the roll moment. Additionally, for the given test dataset, the ANNs trained under the SMMO strategy show a smaller error in predicting the pitching moment compared to those trained with the MMSO strategy, though this does not extend to the rolling moment.

Table 16. MMSO and SMMO test dataset RMSE.

	C_l-MMSO	C_l-SMMO	C_m-MMSO	C_m-SMMO
Incompressible	5.508×10^{-3}	5.942×10^{-3}	2.72×10^{-3}	2.716×10^{-3}
Compressible	3.202×10^{-3}	3.582×10^{-3}	2.61×10^{-3}	2.42×10^{-3}
Joint	3.59×10^{-3}	3.975×10^{-3}	2.625×10^{-3}	2.461×10^{-3}

5. Discussion

Both methodologies, namely the single model with multiple outputs and the multiple models with a single output, showed excellent agreement with the experimental measurements. However, in contrast to Ross et al. [8], the surrogate model that simultaneously predicted the pitching and rolling moments outperformed the model that made individual predictions. Despite this, some predictions fell outside the tolerance margins. This issue was more pronounced for the rolling moment coefficient than for the pitching moment coefficient, which is not entirely unexpected given the higher accuracy observed for the C_m predictions.

In Section 4 we discussed that the wind tunnel pitching moment measurements exhibit asymmetry for some sideslip angle polars and symmetric SACCON configurations. However, this discrepancy could be attributed to tolerances in the wind tunnel measurements. Because of these tolerance errors, it is not possible to determine whether the moment coefficients are truly symmetrical or not. The same issue was also observed for the rolling moment. To address this, we explored a data augmentation technique. Assuming complete symmetry for the aircraft geometry and its moment response, we mirrored the polars with non-zero sideslip angles measurements, resulting in an additional set of *symmetric* polars. Duplicated entries were replaced by their mean values to enforce complete symmetry within the augmented dataset. We expected the neural network to recognize this underlying symmetry. However, our evaluation showed a decrease in accuracy and compliance with tolerances compared to the previously presented results. One crucial factor influencing the algorithm's performance is the amount of data available for training. Both insufficient or excessive amounts of data can be detrimental to the model. In the former scenario, the algorithms struggle to learn as they only have access to a few samples, while in the latter, the neural network may become susceptible to overfitting. In this case, the exact cause of the performance decline is unclear.

Additionally, we observed deviations in the rolling moment coefficient ($C_l \neq 0$) for certain angle of attack polars depicting a fully symmetric aircraft configuration. This deviation is likely due to noisy measurements and/or, as previously mentioned regarding the asymmetry measurements, to intrinsic wind tunnel measurement tolerances. We intentionally did not address this issue before training the machine learning models to allow for experimentation with the given dataset.

Therefore, future work could focus on the treatment and preprocessing of both moment coefficients. We strongly consider that implementing preprocessing filters to denoise the rolling moment measurements may enhance the data quality and, consequently, improve the results. Additionally, symmetry for both moment coefficients could be enforced through data augmentation, as described earlier, or directly within the machine learning architecture, as outlined in the work by Otto et al. [33].

The current approach for computing derivatives required for the k-fold loss function relies on finite differences, as they are judged to be the most robust way of assessing derivatives from experimental data within the industry. Therefore, finite differences were employed across all models to ensure a consistent basis for comparison. However, in artificial neural networks, derivatives can be accessed using backpropagation, which presents an interesting consideration for improving the current framework.

6. Conclusions

Using feed-forward neural networks, we constructed nonlinear response surfaces for the pitching and rolling moment coefficients of a combat aircraft, relying solely on

wind tunnel measurements. A thorough analysis of the results highlights the superior performance of artificial neural networks compared to ensemble learning techniques in creating surrogate models for the given aerodynamic data. Furthermore, the simultaneous prediction of multiple aerodynamic coefficients showed increased accuracy compared to predicting each coefficient individually.

The developed strategy, involving the usage of k-fold cross-validation, allowed us to fine-tune the hyperparameters of the neural networks. Additionally, this method helped addressing the existing challenge of incorporating design tolerances into the design process. By minimizing an appropriate loss function, we prioritize models that achieve a favorable trade-off between high prediction accuracy and high tolerance compliance. Thus, a model that has a slightly higher RMSE but a significantly greater number of predictions within the tolerances can exhibit a lower overall loss, indicating its superior performance according to our evaluation criteria.

The neural networks' predictions for unseen data highlight the accuracy of machine learning as nonlinear surrogate models. Additionally, the proposed network can predict aerodynamic coefficients within milliseconds, representing a substantial improvement in speed and cost-effectiveness compared to traditional methods. This advancement holds significant promise, encouraging the integration of machine learning models in the industrial sector.

Author Contributions: Conceptualization, G.S., H.-J.S., M.S. and D.N.; methodology, G.S.; software, G.S.; resources, D.N. and N.R.G.; writing—original draft preparation, G.S.; writing—review and editing, H.-J.S. and D.N.; supervision, E.Ö. and N.R.G.; project administration, G.S., E.Ö. and D.N.; funding acquisition, N.R.G. All authors have read and agreed to the published version of the manuscript.

Funding: The funding of the presented work within the Luftfahrtforschungsprogramm VI-1 (LUFO VI-1) project DIGIfly (FKZ: 20X1909G and 20X1909C) by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) is gratefully acknowledged.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to confidentiality.

Acknowledgments: We would like to thank Andreas Schütte for the pictures permission as well as the SACCON data.

Conflicts of Interest: Authors Hans-Jörg Steiner, Michael Schäfer and David Naumann were employed by the company Airbus Defense and Space (AD&S). The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Bhatnagar, S.; Afshar, Y.; Pan, S.; Duraisamy, K.; Kaushik, S. Prediction of aerodynamic flow fields using convolutional neural networks. *Comput. Mech.* **2019**, *64*, 525–545. [CrossRef]
2. Eivazi, H.; Tahani, M.; Schlatter, P.; Vinuesa, R. Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations. *Phys. Fluids* **2022**, *34*, 075117. [CrossRef]
3. Zahn, R.; Winter, M.; Zieher, M.; Breitsamter, C. Application of a long short-term memory neural network for modeling transonic buffet aerodynamics. *Aerosp. Sci. Technol.* **2021**, *113*, 106652. [CrossRef]
4. Andrés-Pérez, E. Data Mining and Machine Learning Techniques for Aerodynamic Databases: Introduction, Methodology and Potential Benefits. *Energies* **2020**, *13*, 5807. [CrossRef]
5. Andres, E.; Paulete-Periáñez, C. On the application of surrogate regression models for aerodynamic coefficient prediction. *Complex Intell. Syst.* **2021**, *7*, 1991–2021. [CrossRef]
6. Teimourian, A.; Rohacs, D.; Dimililer, K.; Teimourian, H.; Yildiz, M.; Kale, U. Airfoil aerodynamic performance prediction using machine learning and surrogate modeling. *Heliyon* **2024**, *10*, e29377. [CrossRef]
7. Yetkin, S.; Abuhanieh, S.; Yigit, S. Investigation on the abilities of different artificial intelligence methods to predict the aerodynamic coefficients. *Expert Syst. Appl.* **2024**, *237*, 121324. [CrossRef]
8. Ross, J.; Jorgenson, C.; Nørgaard, M. *Reducing Wind Tunnel Data Requirements Using Neural Networks*; NASA Technical Memorandum 112193; NASA: Washington, DC, USA, 1997; 16p. Available online: https://ntrs.nasa.gov/api/citations/19970021749/downloads/19970021749.pdf (accessed on 18 July 2024).

9. Karali, H.; Demirezen, M.U.; Yukselen, M.A.; Inalhan, G. Design of a Deep Learning Based Nonlinear Aerodynamic Surrogate Model for UAVs. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020. [CrossRef]
10. Patri, A.; Patnaik, Y. Random Forest and Stochastic Gradient Tree Boosting Based Approach for the Prediction of Airfoil Self-noise. *Procedia Comput. Sci.* **2015**, *46*, 109–121. [CrossRef]
11. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; Adaptive Computation and Machine Learning; The MIT Press: Cambridge, MA, USA; London, UK, 2016.
12. Etkin, B.; Reid, L. *Dynamics of Flight: Stability and Control*; Wiley: Hoboken, NJ, USA, 1995.
13. Schütte, A.; Huber, K.C.; Frink, N.T.; Boelens, O.J. Stability and Control Investigations of Generic 53 Degree Swept Wing with Control Surfaces. *J. Aircr.* **2018**, *55*, 502–533. [CrossRef]
14. Cummings, R.; Schütte, A. The NATO STO task group AVT-201 on 'extended assessment of stability and control prediction methods for NATO air vehicles'. In Proceedings of the 32nd AIAA Applied Aerodynamics Conference, Atlanta, GA, USA, 16–20 June 2014. [CrossRef]
15. Vicroy, D.D.; Loeser, T.D.; Schütte, A. Static and Forced-Oscillation Tests of a Generic Unmanned Combat Air Vehicle. *J. Aircr.* **2012**, *49*, 1558–1583. [CrossRef]
16. Schütte, A.; Hummel, D.; Hitzel, S.M. Flow Physics Analyses of a Generic Unmanned Combat Aerial Vehicle Configuration. *J. Aircr.* **2012**, *49*, 1638–1651. [CrossRef]
17. Vicroy, D.D.; Huber, K.C.; Schütte, A.; Rein, M.; Irving, J.P.; Rigby, G.; Löser, T.; Hübner, A.R.; Birch, T.J. Experimental Investigations of a Generic Swept Unmanned Combat Air Vehicle with Controls. *J. Aircr.* **2018**, *55*, 475–501. [CrossRef]
18. Rajkumar, T.; Aragon, C.; Bardina, J.; Britten, R. Prediction of Aerodynamic Coefficients for Wind Tunnel Data Using a Genetic Algorithm Optimized Neural Network. 2002. Available online: https://ntrs.nasa.gov/citations/20020094296 (accessed on 18 July 2024).
19. Shwartz-Ziv, R.; Armon, A. Tabular data: Deep learning is not all you need. *Inf. Fusion* **2022**, *81*, 84–90. [CrossRef]
20. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
21. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, New York, NY, USA, 13–17 August 2016; pp. 785–794. [CrossRef]
22. Schapire, R.E. A brief introduction to boosting. In Proceedings of the 16th International Joint Conference on Artificial Intelligence—Volume 2, IJCAI'99, San Francisco, CA, USA, 31 July–6 August 1999; pp. 1401–1406.
23. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
24. Maćkiewicz, A.; Ratajczak, W. Principal components analysis (PCA). *Comput. Geosci.* **1993**, *19*, 303–342. [CrossRef]
25. Geladi, P.; Kowalski, B.R. Partial least-squares regression: A tutorial. *Anal. Chim. Acta* **1986**, *185*, 1–17. [CrossRef]
26. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
27. Russell, S.J.; Norvig, P.; Davis, E. *Artificial Intelligence: A Modern Approach, Global Edition*, 3rd ed.; Prentice Hall Series in Artificial Intelligence; Pearson: Boston, MA, USA, 2016.
28. Segal, M.; Xiao, Y. Multivariate random forests. *WIREs Data Min. Knowl. Discov.* **2011**, *1*, 80–87. [CrossRef]
29. Sener, O.; Koltun, V. Multi-Task Learning as Multi-Objective Optimization. *arXiv* **2019**, arXiv:1810.04650.
30. Song, G.; Chai, W. Collaborative Learning for Deep Neural Networks. *arXiv* **2018**, arXiv:1805.11761.
31. Mordan, T.; Thome, N.; Henaff, G.; Cord, M. Revisiting Multi-Task Learning with ROCK: A Deep Residual Auxiliary Block for Visual Detection. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31.
32. Volpiani, P.S. Are random forests better suited than neural networks to augment RANS turbulence models? *Int. J. Heat Fluid Flow* **2024**, *107*, 109348. [CrossRef]
33. Otto, S.E.; Zolman, N.; Kutz, J.N.; Brunton, S.L. A Unified Framework to Enforce, Discover, and Promote Symmetry in Machine Learning. *arXiv* **2023**, arXiv: 2311.00212.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.