The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

DATA-DRIVEN REDUCED ORDER MODELING FOR AERODYNAMIC FLOW PREDICTIONS ECCOMAS CONGRESS 2022

D.A Hines Chaves[1] and P. Bekemeyer[2]

[1,2] German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, $C^2A^2S^2E$
Lilienthalplatz 7, 38108 Braunschweig, Germany

[1] derrick.hineschaves@dlr.de, [2] philipp.bekemeyer@dlr.de

Key words: Reduced-order model, Deep Learning, Proper Orthogonal Decomposition, Multi-layer Perceptron, Autoencoder, Aerodynamics

Abstract. During each aircraft program a vast amount of aerodynamics data has to be generated to judge performance, structural loads as well as handling qualities. Within the past years the usage of computational fluid dynamics has significantly increased providing accurate insights into aircraft behaviour at early design stages and therefore at least partially enabled the mitigation of costly design changes. However, fully relying on high fidelity aerodynamic data is still computational prohibitive. Hence, data-driven models have gained an increasing attention in recent years. These methods not only provide continuous models but also enable the inclusion of highly accurate aerodynamic results in time-critical environments. This paper aims at applying deep learning techniques to derive such models and compare them to state of the art reduced order modeling techniques. In particular, three deep learning methods, a Multi-layer perceptron for distribution predictions, a Multi-layer perceptron for pointwise predictions and an Autoencoder coupled with an interpolation technique are compared to Proper Orthogonal Decomposition and Isomap with latent space interpolation. For all methods an efficient methodology to determine hyperparameters is outlined and applied. Results are presented for an Airbus provided XRF1 dataset which includes surface pressure distributions at various Mach numbers and angles of attack.

1 Introduction

Aerodynamic data, no matter if performance values, handling qualities or loads, are an integral part of every aircraft design, optimization or certification program. The sheer amount of data needed typically results in relying on different data sources as well as different fidelity levels. However, continuous models are generally desirable to interact with other disciplines and drive design activities. Hence, data-driven models have been proposed to not only yield such models but also to enable the integration of data from various sources and fidelities. Besides long-standing conventional models, especially deep-learning approaches have gathered significant interest in the aerodynamics community [1]. Nevertheless, even though several well-established deep-learning libraries are freely available, their applicability for physics-based problems on an

industrial level remains unclear. Moreover, rigorous comparison to conventional techniques are only partially available at best [2]. Therefore, the GARTEUR group "Machine learning and data-driven approaches for aerodynamic analysis and uncertainty quantification" was initiated to bring industrial and academic experts together to evaluate the potential of emerging methods.

When looking at classical reduced order models one can split them in two classes, intrusive and non-intrusive approaches. The former have shown highly accurate results but they require access to the underlying solver also during the evaluation stage. The latter provide a black-box representation of the problem enabling rapid turn-around times and a straightforward inclusion into other, complex workflows. Hence, within this work we focus on non-intrusive, often also called data-driven, approaches. Within this class proper orthogonal decomposition (POD) [3] as a dimensionality reduction technique combined with an interpolation method such as radial basis functions or Gaussian Processes is arguably the most common method. Application examples for aerodynamics are widespread and can be found in [4, 5, 6, 7, 8]. Models are reported to be easy to construct and highly accurate as long as no nonlinear behaviour, which violates the underlying POD assumption, is present. An alternative to POD is the nonlinear dimensionality reduction technique Isomap [9] which is based on multidimensional scaling. Just like POD it can be employed to extract low-dimensional structures from a high-dimensional sampling set and coupled with an interpolation method to predict solutions at previously untried conditions [6].

Recently Neural Networks (NN) as an alternative to classical reduced order models became popular since deep learning techniques have proven successful in the extraction and representation of hierarchical data features [10]. Publications are available for the prediction of aerodynamic coefficients for airfoils including integral quantities (lift and drag) [11], fields (surface pressure distribution) [12] as well as unsteady forces [13, 14]. In [15], Convolutional Neural Networks are introduced to predict the velocity field in non-uniform steady laminar flows while several extensions are available in [16, 17, 18]. In [19] an application accounting for changing airfoil geometries has been presented employing Generative Adversarial Networks. A first extension towards an industrial relevant 3D case featuring transonic flows including shocks and boundary layer separation has been published in [2] relying on a multi-layer perceptron which is used for the pointwise prediction of pressure coefficients. The aforementioned paper also includes an initial comparison to well-established reduced order methods.

In this paper we focus on the comparison of three deep learning methodologies with classical reduced order models for the prediction of wing surface pressure distributions. This task is highly relevant for industrial applications since accurately predicting the pressure coefficients over the wing is essential for the determination of aerodynamic loads, transition prediction and inverse design. Section 2 provides background information on all applied modeling techniques. Section 3 outlines the model selection techniques that are employed during the experiments. In Section 4 we present the test case, based on the XRF1 configuration provided by Airbus and subsequently discuss results in Section 5.

2 DATA-DRIVEN METHODS FOR AERODYNAMIC FLOW PREDICTIONS

In this section we present the five data-driven methods that we will use for the prediction of distributed aerodynamic quantities. We have available a point cloud with n points in space corresponding to the nodes of a CFD mesh on the underlying geometry. The coordinate matrix

$C \in \mathbb{R}^{3,n}$ is given by

$$C = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n] \tag{1}$$

where each $\mathbf{c}_j \in \mathbb{R}^3$ is the location of point j. In addition, we consider m full order solutions $Y \in \mathbb{R}^{n,m}$ corresponding to m parameter combinations $P \in \mathbb{R}^{p,m}$

$$Y = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m] \tag{2}$$
$$P = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m] \tag{3}$$

where each snapshot $\mathbf{y}_i \in \mathbb{R}^n$ corresponds to the distributed aerodynamic quantity over all n points for the parameter configuration $\mathbf{p}_i \in \mathbb{R}^p$. The full order model (FOM) is assumed to be given by

$$\mathbf{y} = f(\mathbf{p}) \tag{4}$$

The goal of the data-driven methods is to approximate f via a surrogate function $\hat{f}$ so that

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{p}) \approx f(\mathbf{p}) \tag{5}$$

Hence, for new parameter combinations a prediction can be computed significantly faster than with the full order model.

As for the experiments in this paper, the snapshots are consistently split into three datasets as follows:

1. Train: m_{train} snapshots Y_{train} that are used for training the models.
2. Validation: m_{val} snapshots Y_{val} that are used for choosing the best hyperparameters for the model.
3. Test: m_{test} snapshots Y_{test} that are used for testing the models.

This provides a common setting for the comparison among models. The model selection techniques based on the performance on the validation samples are described in Section 3, while Section 5 discusses the results with respect to the test samples.

2.1 Proper orthogonal decomposition

A POD orthogonal basis $\Phi \in \mathbb{R}^{n,d}$ with $d \leq m$ modes is obtained using the method of snapshots [20]. These linear combinations of the snapshots Y also yield a lower dimensional representation $Z \in \mathbb{R}^{d,m}$ of the snapshots where

$$Y \approx \Phi Z \tag{6}$$
$$\Phi \approx Y Z^\top \tag{7}$$
$$\Phi = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_d] \tag{8}$$
$$Z = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m] \tag{9}$$

and each $\mathbf{z}_i \in \mathbb{R}^d$ corresponds to the lower d-dimensional representation of $\mathbf{y}_i$. The matrix $\hat{Y} := \Phi Z$ is the best approximation of rank d of the snapshot matrix Y with respect to least squares [21].

Once Φ and Z are computed, the POD model can be coupled with an interpolation technique to predict solutions for new parameters $\mathbf{p}$. The reconstructions for the the known parameters in P are linear combinations of the POD basis.

$$\hat{\mathbf{y}}(\mathbf{p}_i) = \Phi \mathbf{z}_i \tag{10}$$

For new parameter combinations the prediction is also computed as a linear combination of the POD basis. The coefficients are determined by an interpolation function $\hat{\mathbf{z}}\colon \mathbb{R}^p \to \mathbb{R}^d$. As for the interpolation we consider a radial basis kernel $\psi\colon \mathbb{R} \to \mathbb{R}$, such as the linear kernel $\psi(r) = r$, or the TPS kernel $\psi(r) = r^2 log(r)$ [22]. Considering $\Psi\colon \mathbb{R}^p \to \mathbb{R}^m$ as

$$\Psi(\mathbf{p}) = \begin{bmatrix} \psi(||\mathbf{p} - \mathbf{p}_1||) \\ \vdots \\ \psi(||\mathbf{p} - \mathbf{p}_m||) \end{bmatrix} \tag{11}$$

the resulting interpolation is expressed in matrix notation as

$$\hat{\mathbf{z}}(\mathbf{p}) = W\Psi(\mathbf{p}) \tag{12}$$

where $W \in \mathbb{R}^{d,m}$ is a weight matrix to be calculated. Finally the prediction $\hat{\mathbf{y}}(\mathbf{p})$ is computed as a linear combination of the POD basis with coefficients $\hat{\mathbf{z}}(\mathbf{p})$.

$$\hat{\mathbf{y}}(\mathbf{p}) = \Phi\hat{\mathbf{z}}(\mathbf{p}) \tag{13}$$

Moreover, the model may be extended to include regularization techniques and augmentation with additional polynomial terms of $\mathbf{p}$ in equation 12. We refer to the resulting model as POD+I.

2.2 Isomap

Isomap [9] is a nonlinear dimensionality reduction technique that attempts to preserve the geodesic distances between all pair of points with respect to an underlying data manifold [6]. It yields a nonlinear mapping from the high dimensional space to the low dimensional space for a fixed set of snapshots. Similarly as for POD, interpolation techniques can be used to determine a low dimensional representation $\hat{\mathbf{z}}(\mathbf{p})$ for new parameters $\mathbf{p}$. Then a non-parametric back-mapping is used to make the prediction as a weighted linear combination of the snapshots $\mathbf{y}_i$ corresponding to the N nearest neighbors of $\hat{\mathbf{z}}(\mathbf{p})$. We refer to the resulting model as Isomap+I and use the implementation from [6].

2.3 Multi-layer perceptron decoder

A Multi-layer perceptron (MLP) with L hidden layers can be used to predict the distributed quantity over all n points. Here the input is $\mathbf{a}^{[0]} := \mathbf{p} \in \mathbb{R}^p$. At each hidden layer $1 \leq l \leq L$ a linear transformation is applied followed by a nonlinear activation.

$$\mathbf{z}^{[l]} = W^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]} \tag{14}$$

$$\mathbf{a}^{[l]} = \sigma^{[l]}(\mathbf{z}^{[l]}) \tag{15}$$

where at layer l, $W^{[l]} \in \mathbb{R}^{n^{[l]},n^{[l-1]}}$ is the weight matrix , $\mathbf{b}^{[l]} \in \mathbb{R}^{n^{[l]}}$ is the bias vector, $n^{[l]}$ is the number of neurons or dimension and $\sigma^{[l]}$ is a nonlinear activation function such as $ReLU$. The prediction at the final layer L is computed as

$$\hat{\mathbf{y}}(\mathbf{p}) = W^{[L+1]}\mathbf{a}^{[L]} + \mathbf{b}^{[L+1]} \tag{16}$$

where $W^{[L+1]} \in \mathbb{R}^{n,n^{[L]}}$. Hence the predictions are computed as linear combinations of the $n^{[L]}$ columns in $W^{[L+1]}$ plus a bias vector. In this setting the dimension of the last hidden layer is given by $n^{[L]}$. The dimension of the previous hidden layers $1 \leq l < L$ is determined using a downsampling multiplier $0 < r \leq 1.0$, so that the dimension of each previous hidden layer is reduced consecutively by the multiplicative factor r as follows:

$$n^{[l]} = \lfloor n^{[l+1]} r \rceil \tag{17}$$

where $\lfloor x \rceil$ denotes the nearest integer of x. We refer to this model as MLP-Decoder. The structure of this model is illustrated in Figure 1.

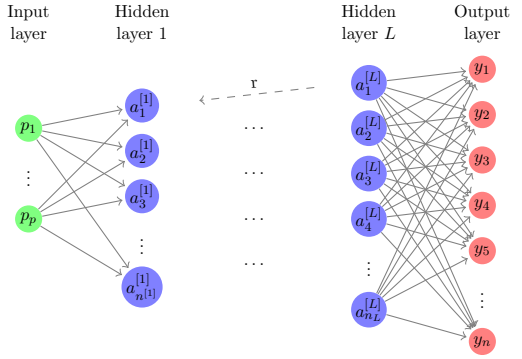


Figure 1: MLP decoder

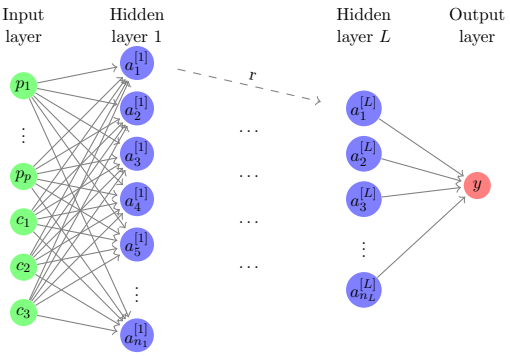


Figure 2: MLP pointwise

2.4 Multi-layer perceptron pointwise

Alternatively, one can use a Multi-layer perceptron to predict the quantity of interest at a particular point. The input is then given by the parameters $\mathbf{p}$ together with the $3D$ coordinates $\mathbf{c}$ of the point. That is, the input is $\mathbf{a}^{[0]} = \mathbf{p} \oplus \mathbf{c} \in \mathbb{R}^{p+3}$ where $\oplus$ represents the concatenation of vectors. The output $MLP(\mathbf{p} \oplus \mathbf{c})$ is the one-dimensional prediction of the quantity of interest at the location specified by $\mathbf{c}$ for the parameters $\mathbf{p}$. The prediction of the distributed quantity over all mesh nodes is then calculated as

$$\hat{\mathbf{y}}(\mathbf{p}) = \begin{bmatrix} MLP(\mathbf{p} \oplus \mathbf{c}_1) \\ \vdots \\ MLP(\mathbf{p} \oplus \mathbf{c}_n) \end{bmatrix} \tag{18}$$

For this model the dimension of the first hidden layer is given by $n^{[1]}$. Following [2] the dimension of the next hidden layers $1 < l \leq L$ is determined using a downsampling multiplier $0 < r \leq 1.0$. In this way the dimension of each next hidden layer is reduced consecutively by the multiplicative factor r as expressed by:

$$n^{[l]} = \lfloor n^{[l-1]} r \rceil \tag{19}$$

where $\lfloor x \rceil$ denotes the nearest integer of x. We refer to this model as MLP-Pointwise. The structure of this model is shown in Figure 2.

2.5 Autoencoder

An autoencoder (AE) is a neural network that is trained to transfer its input to its output and in doing so learns lower-dimensional feature representations from unlabeled high dimensional data [23, 24]. The autoencoder may be seen as a combination of two neural networks: an encoder $h\colon \mathbb{R}^n \to \mathbb{R}^d$ and a decoder $g\colon \mathbb{R}^d \to \mathbb{R}^n$. The encoder h maps the data $\mathbf{y}$ to a low dimensional latent representation $\mathbf{z} = h(\mathbf{y})$. The decoder g maps back the latent representation $\mathbf{z}$ to a reconstruction $\hat{\mathbf{y}} = g(\mathbf{z})$. Here the input is $\mathbf{a}^{[0]} = \mathbf{y} \in \mathbb{R}^n$ and the output is the reconstruction $\hat{\mathbf{y}} \in \mathbb{R}^n$. The dimension of the first hidden layer of the encoder is given by $n^{[1]}$. The dimension of the next hidden layers $1 < l \leq L$ is determined using a downsampling multiplier $0 < r \leq 1.0$. In this way the dimension of each next hidden layer is reduced consecutively by the multiplicative factor r as in the case of MLP-Pointwise. The decoder's structure is a mirror of the encoder as illustrated in Figure 3. Similarly as for POD+I, the autoencoder is coupled with an interpolation function $\hat{\mathbf{z}}\colon \mathbb{R}^p \to \mathbb{R}^d$. Altogether the prediction of the model for a new parameter combination $\mathbf{p}$ is the reconstruction that the decoder g makes from the interpolated latent representation $\hat{\mathbf{z}}(\mathbf{p})$ as expressed by:

$$\hat{\mathbf{y}}(\mathbf{p}) = g(\hat{\mathbf{z}}(\mathbf{p})) \tag{20}$$

We refer to this model as AE+I.

3 MODEL SELECTION

This section outlines the two methodologies used to choose the hyperparameters of the models. The selection is done based on the mean absolute error (MAE) with respect to the validation samples, which is calculated as indicated by equation 21.

$$MAE(\hat{Y}_{val}; Y_{val}) = \frac{1}{m_{val} \cdot n} \left(\sum_{i=1}^{m_{val}} \sum_{j=1}^{n} |\hat{\mathbf{y}}_{i;j} - \mathbf{y}_{i;j}| \right) \tag{21}$$

The two approaches used to find the best hyperparameters are the following:

1. Full Factorial Grid Search:
   For POD+I, Isomap+I, and AE+I models, the hyperparameters of the interpolation are changed following a full factorial design as all options are discrete values. For each combination of hyperparameters, the validation MAE is obtained and the combination of hyperparameters leading to the lowest MAE is selected. The hyperparameters for the interpolation are shown in the appendix in Table 4.

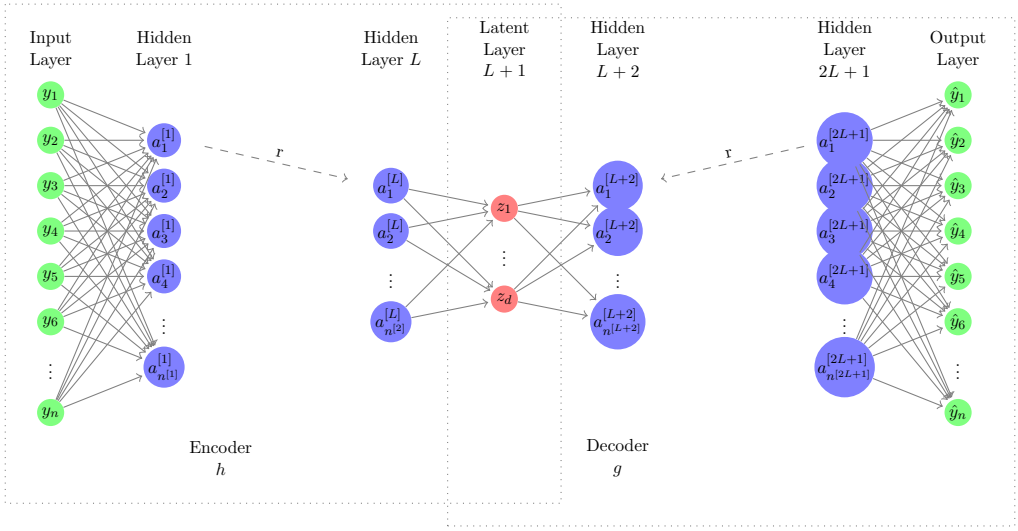


Figure 3: Autoencoder

2. Surrogate-Based Optimization:
   The hyperparameters of the NN models (AE, MLP-Decoder and MLP-Pointwise) are optimized using a Bayesian optimization strategy known as Surrogate Based Optimization. It is an advantageous approach for the optimization of black-box problems as it notably reduces the number of function evaluations by substituting the black box with a meta-model [25]. The optimization consists of an initial Design of Experiments stage (DOE, 15 iterations), followed by the Expected Improvement Infill Criteria (EI, 10 iterations) and a final stage of Expected Improvement Infill Criteria within a moving Trust Region (EI-TR, 5 iterations). The objective function for MLP-Decoder and MLP-Pointwise is the mean absolute error computed on the validation set. For AE the objective function is the mean squared error with respect to the training data.

4 TEST CASE

The test case configuration considered here is an industrial-relevant aircraft configuration known as XRF1. The XRF1 is an Airbus provided industrial standard multi-disciplinary research test case representing a typical configuration for a long range wide body aircraft. It is used by Airbus to engage with external partners on development and demonstration of relevant capabilities and technologies. Computational data has been provided from Airbus based on the XRF1 half-body configuration. High-fidelity RANS-CFD simulations were carried out with the DLR flow solver TAU [26], using the SST turbulence model. Structural deformations are accounted for by coupling the CFD analysis to a computational structural mechanics investigation which relies on a finite element model. Afterwards all results are projected onto the same aircraft surface mesh for which only the wing surface is retained. The corresponding surface grid consists of $n = 110,169$ grid points and the point could is shown in Figure 4. Note that, the surface mesh connectivity for this particular mesh is not available. Hence, certain approaches relying on connectivity information such as graph convolutional networks [27] are not applicable. A total of $m = 89$ pressure coefficient (c_p) distributions were computed for four different Mach

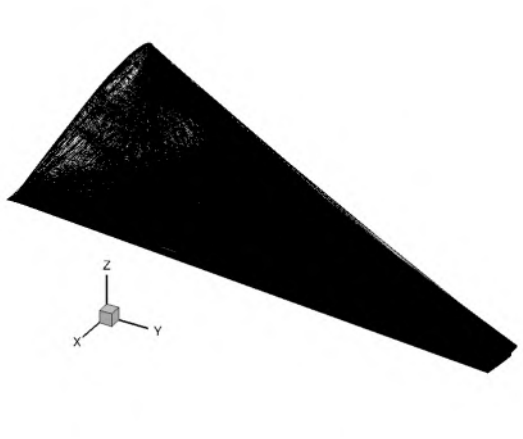


Figure 4: Configuration

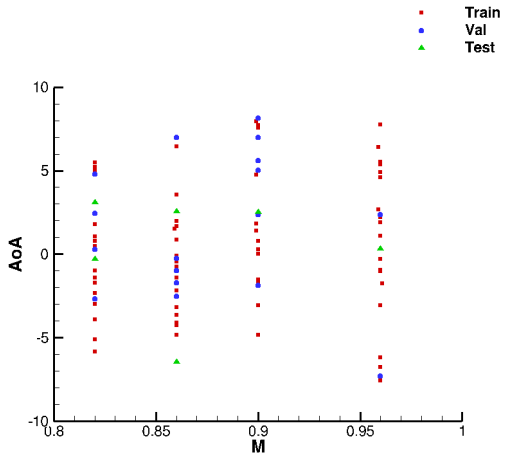


Figure 5: Dataset split

Table 1: Description of the test case

Surface Points	Mach Range	AoA Range	Train Samples	Val. Samples	Test Samples
110169	$[0.82, 0.96]$	$[-7.6°, 8.1°]$	66	17	6

numbers $0.82, 0.86, 0.90$ and 0.96 and angles of attack in the range of $[-7.6°, 8.1°]$. The sampling points are displayed in Figure 5. The Reynolds number was fixed to $Re = 25 \cdot 10^6$. Out of the $m = 89$ total samples 6 are selected for testing, 17, approximately twenty percent, are selected for validation while the remaining 66 samples are used for training. This partition of the data is pictured in Figure 5. Table 1 contains a summary of this test case.

The reduced order modeling methods are part of the DLR Surrogate Modeling for AeRo data Toolbox in python (SMARTy) [28]. SMARTy is a toolbox for predictive modeling of aerodynamic scalar or high-dimensional quantities using state-of-the-art data-driven methods. It allows in a very user-friendly way the construction of reduced order models from CFD data in a unified layer that includes several preprocessing and postprocessing options, training of neural networks with PyTorch [29] and TensorFlow [30] as backends and surrogate-based hyperparameter optimization. In addition, it features a model selection module that provides a grid search functionality which automatically takes care of building and training various models and selecting the best one. For the deep learning models in this paper we use TensorFlow as the backend.

5 RESULTS

This section discusses the results obtained predicting pressure coefficient distributions at various combinations of Mach number and angle of attack for the test case presented in the previous section. First, we present the results from the hyperparameter optimization. Then, we

analyze the performance of the models with respect to the testing samples. We conclude with a discussion regarding the computational cost of all models.

5.1 Surrogate-based optimization

As presented in Section 3 the hyperparameters of MLP-Decoder, MLP-Pointwise and AE+I were optimized using surrogate-based optimization. For this a Kriging-based surrogate model was employed in order to choose the optimal hyperparameters for each NN. For this the construction and training of a particular NN model can be seen as a black-box function. For each of the evaluations of the black-box objective function (NN training), the weights of the neural network were optimized using the well-known optimizer Adam [31].

For the three models the optimizable hyperparameters were the learning rate, the number of hidden layers L, and the downsampling multiplier r. The other hyperparameters were fixed based on previous experience as well as running time and model capacity constraints. The mean squared error was used as the loss function and early stopping with a window size of 10 was employed in order to avoid overfitting. As for the activation function, all models used $ReLU$.

In the case of MLP-Decoder the last hidden dimension $n^{[L]}$ was fixed at 256 and the batch size at 16. It was trained for a maximum of 400 epochs. As for MLP-Pointwise, the first hidden dimension was fixed at 256 and the batch size at 4098. It was trained for maximum of 250 epochs. Concerning AE+I, the first hidden dimension was fixed at 256, the latent dimension at 64 and the batch size at 16. It was trained for maximum of 150 epochs. For MLP-Decoder and AE+I the batch size refers to the number of snapshots, while for MLP-Pointwise it represents the number of parameter-coordinate pairs.

The optimization of neural networks is known to be heavily dependent on the chosen learning rate and its order of magnitude. Thus this hyperparameter was logarithmically scaled to allow for a suitable search across different orders of magnitude. The optimal learning rate for MLP-Decoder was found to be 7.50×10^{-3}, while for MLP-Pointwise and for AE+I it was an order of magnitude lower, 5.62×10^{-4} and 1.64×10^{-4} respectively.

The optimal MLP-Decoder consists of 8 hidden layers and a downsampling multiplier of 0.838. As for MLP-Pointwise, the best model features 10 hidden layers with a downsampling multiplier of 0.906. Regarding AE+I, the best hyperparameters are 2 hidden layers in the encoder and a downsampling multiplier of 0.7. All optimal hyperparameters and the dimensions of the layers are shown in the appendix in Tables 6, 7, 8 and 9.

5.2 Interpolation grid search

For POD+I, Isomap+I and AE+I the hyperparameters of the interpolation were chosen by performing a grid search over the options displayed in Table 4 in the appendix. The best interpolation technique for POD+I and AE+I was found to be TPS with regularization, while for Isomap+I a simpler linear interpolation yielded the best results. In the case of Isomap+I the latent space was fixed to be two-dimensional and the model chose to use the 16 nearest neighbors for the backward mapping. It is likely that the nonlinear dimensionality reduction step from Isomap allowed the plain linear interpolation to suffice. Regarding scaling, the three models scale the parameters to the unit hypercube, pointing to the usefulness of adequate scaling. All optimal hyperparameters of the interpolation methods are shown in Table 5 in the appendix.

5.3 Results on test samples

In a production environment it is key that the models perform well at new unseen parameter combinations. Accordingly, the performance of the models is assessed using the 6 test samples, which were used neither for training nor validation or hyperparameter selection. The predictive capabilities are evaluated quantitatively using the mean absolute error (MAE), the 95% and 99% quantiles of the absolute error (AE q.95, AE q.99 resp.), the maximum absolute error (Max. AE), the root mean squared error (RMSE), and the r2-score. The maximum error is provided to give a more complete overview of the performances of all models, but one should bear in mind that it is an statistic with very high variance. The quantiles of the absolute error provide a more robust measure of how big the errors get. High errors tend to exist at locations where nonlinear behaviour occurs in the pressure distribution due to the presence of shock waves and flow separation. Hence these quantiles help to better assess the ability of the models to predict local nonlinearities, which is essential for the prediction of loads at transonic flow conditions. In addition, the root mean squared error, penalizes higher errors more the mean absolute error and is thus suitable when high errors are to be avoided.

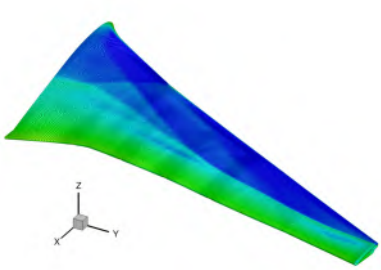
POD+I achieves the lowest MAE among all models, while MLP-Pointwise yields the best performance in terms of all other error metrics. MLP-Pointwise outperforms POD+I by 7.6% and 14.25% in terms of the 95% and 99% error quantiles respectively. This suggests that MLP-Pointwise performs better at capturing localized phenomena such as strong shocks or flow separation. A possible explanation is that the mini-batch stochastic optimization algorithm with a mean squared error loss helps the deep learning methods by forcing them to learn local behaviour and increasing their generalization performance. Moreover, the use of point coordinates provides MLP-Pointwise with additional information that can turn out to be useful for learning such local phenomena by implicitly gathering information on local phenomena and connectivity. All test metrics with respect to the six test samples are shown in Table 2.

Table 2: C_p test metrics

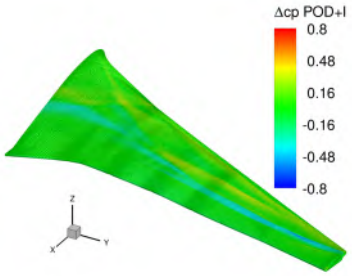
Metric	POD+I	Isomap+I	AE+I	MLP-Dec.	MLP-Pointwise
MAE	0.0293	0.0324	0.0382	0.0362	0.0309
AE. q95	0.1256	0.1424	0.1338	0.1250	0.1163
AE. q99	0.3242	0.3160	0.3047	0.2822	0.2780
Max. AE	1.1824	1.0187	0.9988	1.0603	1.0815
RMSE	0.0748	0.0723	0.0734	0.0706	0.0638
r2-score	0.9659	0.9681	0.9671	0.9696	0.9752

Figures 6 and 7 show a comparison of the prediction errors $\Delta c_p := \hat{\mathbf{y}} - \mathbf{y}$ along with the reference solution for the two test cases that are the farthest away from the training and validation samples. Figure 6 features the case at a Mach number of 0.82 and an angle of attack of 3.105. In this case MLP-Decoder attains the best performance with respect to all metrics. It outperforms POD+I by 26.85% and 8.03% with respect to the 95% and 99% error quantiles, respectively. Figure 7 displays the test case at a Mach number of 0.86 and an angle of attack of

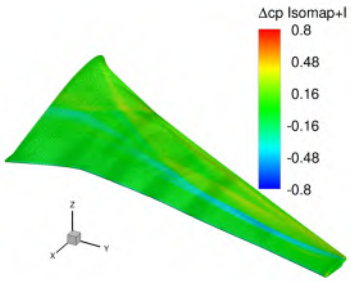
−6.434. This test point is the farthest away from all training samples. For this test case MLP-Pointwise achieves the best performance with respect to all metrics. It outperforms POD+I by 34.80% and 45.20% in terms of the 95% and 99% error quantiles, respectively. This performance appears to support that MLP-Pointwise can better model nonlinear behaviour and extrapolate to conditions in the flight envelope not covered by the training data.



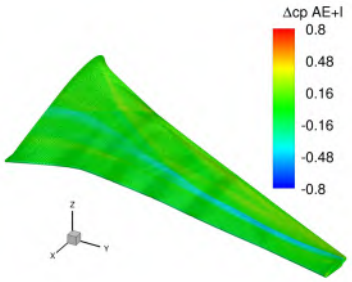
(a) Reference Solution



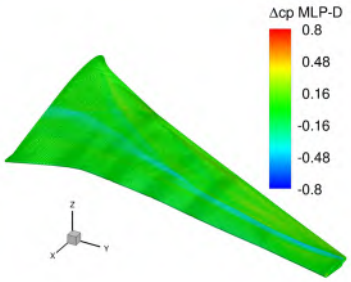
(b) POD+I



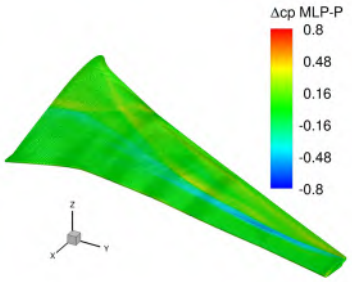
(c) Isomap+I



(d) AE+I

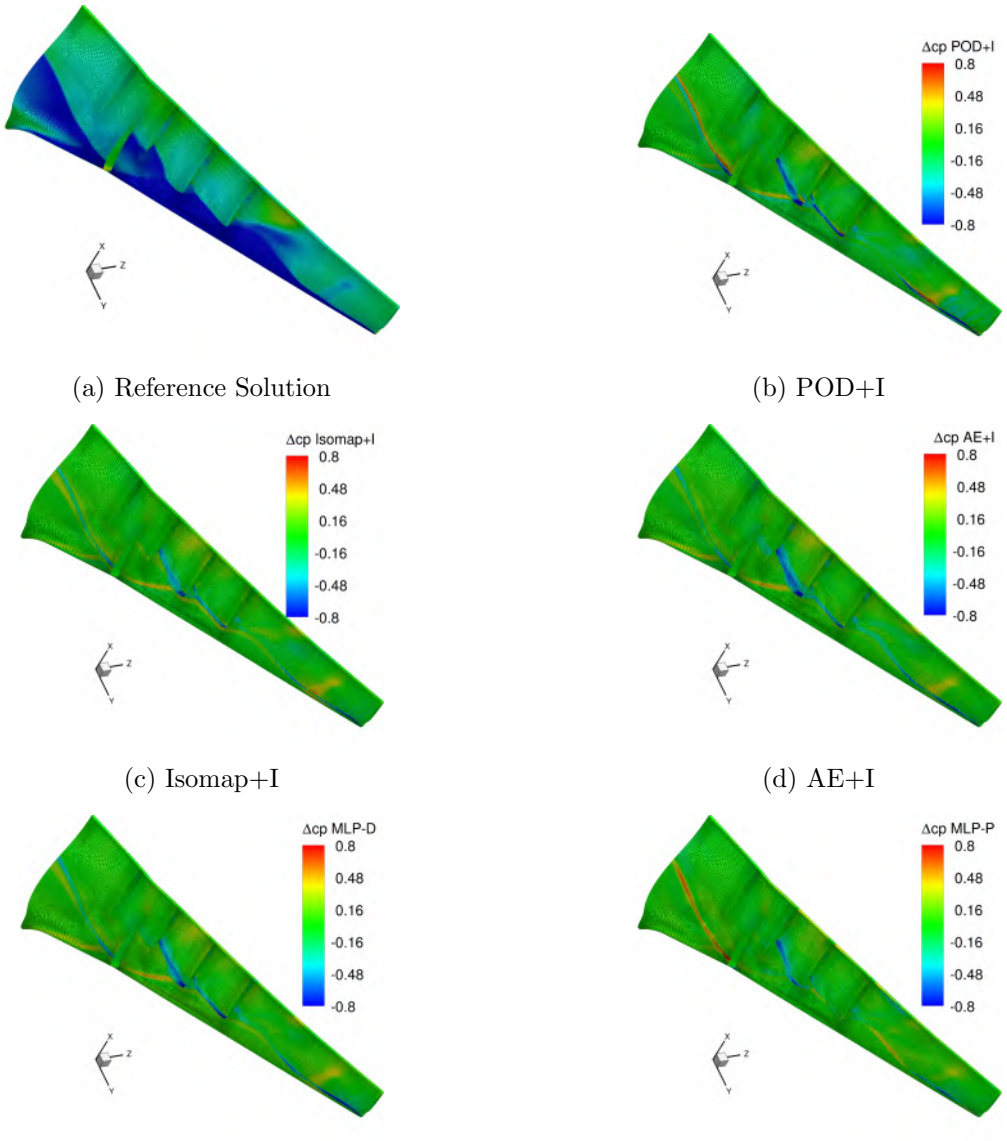


(e) MLP-Decoder



(f) MLP-Pointwise

Figure 6: Comparison between the prediction accuracy for M: 0.82 and AoA: 3.105



(a) Reference Solution

(b) POD+I

(c) Isomap+I

(d) AE+I

(e) MLP-Decoder

(f) MLP-Pointwise

Figure 7: Comparison between the prediction accuracy for M: 0.86 and AoA: -6.434

5.4 Computational cost

For practical applications it is not only important that the data-driven models provide accurate predictions of aerodynamic quantities, but also that they do so in significantly lower time than the time-consuming high fidelity CFD computations. Therefore we provide an overview of the training and evaluation times of the models to help assess the feasibility of their use in time-critical scenarios. MLP-Pointwise was trained using an NVIDIA Quadro P4000 GPU, whereas all other models were trained exclusively on an Intel(R) Xeon(R) W-2135 CPU @ 3.70GHz with

12 cores. POD+I and Isomap+I require the least time to train, 9 and 11 seconds, respectively. For POD+I most of the computation time is required for the regularized TPS interpolation, while for Isomap+I the dimensionality reduction is the most time-consuming step. The training of AE+I and MLP-Decoder is in the order of minutes, each taking 126 and 166 seconds. MLP-Pointwise has a training time in the order of hours, taking 5839 seconds to complete 250 epochs. Regarding the evaluation time, for all models it is significantly under one second and is thus negligible with respect to the CFD simulation time. Table 3 shows for each model the approximate time per epoch, the training time and the time it takes to perform an evaluation of one sample to yield a prediction of the complete pressure coefficient distribution.

Table 3: Approximate computational cost in seconds

	POD+I	Isomap+I	AE+I	MLP-Dec.	MLP-Pointwise
One Epoch	–	–	0.74	0.42	23.84
Training	9	11	126	166	5839
One Evaluation	0.0024	0.0028	0.0126	0.1901	0.2184

6 CONCLUSIONS

In this paper we compared five data-driven methods for the prediction of surface pressure distributions for the XRF1 half configuration. These models provide fast approximations and can be seen as a surrogate for time-consuming RANS-CFD simulations. Proper Orthogonal Decomposition coupled with latent space interpolation (POD+I) was used as the baseline due to its low computational cost and widespread use. POD+I performs well regarding global metrics, however its performance drops when looking at local errors and extrapolating to cases with high local nonlinearities. Isomap coupled with interpolation (Isomap+I) was used as the second baseline and it appears to suffer less from these limitations, likely due to its nonlinear dimensionality reduction technique and local interpolation method.

An Autoencoder coupled with interpolation (AE+I) was used as a hybrid approach between classic reduced order modeling and deep learning. It features a nonlinear dimensionality reduction step as in the case of Isomap. The use of two deep learning approaches without any explicit dimensionality reduction was also investigated. A multilayer perceptron that predicts the surface pressure distribution directly from the parameters (MLP-Decoder) was one of these methods. The other deep learning method was a multilayer perceptron for pointwise predictions (MLP-Pointwise) used to predict the surface pressure coefficient at a given combination of parameters and coordinates. The results suggest that this last method is better suited than the previous ones for capturing local nonlinearities. This method is however the one that requires the most training time. Throughout, all models are capable of evaluating new samples in nearly real-time. This is crucial since it enables their use within time-critical environments.

Given that the main challenges that POD faces are related to locality and nonlinearity, kernel POD coupled with interpolation (KPOD+I) stands out as a natural candidate to improve the performance of the classic POD+I approach. Another possible approach is the use of a neural

network instead of the interpolation technique to form a POD+NN model. Regarding deep learning methods, the best approach employed in this work (MLP-Pointwise) learns and computes predictions point by point without explicitly using information from neighboring points. Graph Neural Networks [27] and Transformers [32] are two state-of-the-art methods that explicitly use neighborhood information during training and prediction time. These methods could learn the spatial correlations with the use of local aggregation schemes and attention-based mechanisms. This should, in turn, further increase the accuracy of the predictions by better capturing nonlinear local phenomena. Thus these methods constitute a promising approach for future work.

Acknowledgment

The authors would like to thank Airbus for providing the XRF1 test case as a mechanism for demonstration of the approaches presented in this paper.

REFERENCES

1. Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine Learning for Fluid Mechanics. *Annual Review of Fluid Mechanics* **52,** 477–508. eprint: https://doi.org/10.1146/annurev-fluid-010719-060214. https://doi.org/10.1146/annurev-fluid-010719-060214 (2020).
2. Campomanes, C. S., Stürmer, P. & Bekemeyer, P. *Fast Predictions of Aircraft Aerodynamics using Deep Learning Techniques* in *AIAA Aviation 2021 Forum* (Aug. 2021).
3. Lucia, D. J., Beran, P. S. & Silva, W. A. Reduced-order modeling: new approaches for computational physics. *Progress in Aerospace Sciences* **40,** 51–117. ISSN: 0376-0421. https://www.sciencedirect.com/science/article/pii/S0376042103001131 (2004).
4. Bui-Thanh, T, Damodaran, M. & Willcox, K. *Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics* in *21st AIAA Applied Aerodynamics Conference* (2003), 4213.
5. Iuliano, E. & Quagliarella, D. Proper orthogonal decomposition, surrogate modelling and evolutionary optimization in aerodynamic design. *Computers & Fluids* **84,** 327–350 (2013).
6. Franz, T., Zimmermann, R., Görtz, S. & Karcher, N. Interpolation-based reduced-order modelling for steady transonic flows via manifold learning. *International Journal of Computational Fluid Dynamics* **28** (Mar. 2014).
7. Fossati, M. Evaluation of aerodynamic loads via reduced-order methodology. *AIAA Journal* **53,** 2389–2405 (2015).
8. Ripepi, M. *et al.* Reduced-order models for aerodynamic applications, loads and MDO. *CEAS Aeronautical Journal* **9,** 171–193 (2018).
9. Tenenbaum, J. B., Silva, V. d. & Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science* **290,** 2319–2323 (2000).
10. Bengio, Y. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning* **2,** 1–127. https://doi.org/10.1561/2200000006 (2009).

11. Santos, M., Mattos, B. & Girardi, R. *Aerodynamic Coefficient Prediction of Airfoils Using Neural Networks* in *46th AIAA Aerospace Sciences Meeting and Exhibit* (American Institute of Aeronautics and Astronautics, Jan. 2008). https://doi.org/10.2514/6.2008-887.
12. Yilmaz, E. & German, B. *A Convolutional Neural Network Approach to Training Predictors for Airfoil Performance* in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (American Institute of Aeronautics and Astronautics, June 2017). https://doi.org/10.2514/6.2017-3660.
13. Kou, J. & Zhang, W. Layered reduced-order models for nonlinear aerodynamics and aeroelasticity. *Journal of Fluids and Structures* **68,** 174–193. ISSN: 0889-9746. https://www.sciencedirect.com/science/article/pii/S0889974616301852 (2017).
14. Rozov, V. & Breitsamter, C. Data-driven prediction of unsteady pressure distributions based on deep learning. *Journal of Fluids and Structures* **104,** 103316. ISSN: 0889-9746. https://www.sciencedirect.com/science/article/pii/S0889974621000992 (2021).
15. Guo, X., Li, W. & Iorio, F. *Convolutional Neural Networks for Steady Flow Approximation* in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, San Francisco, California, USA, 2016), 481–490. ISBN: 9781450342322. https://doi.org/10.1145/2939672.2939738.
16. Jin, X., Cheng, P., Chen, W.-L. & Li, H. Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder. *Physics of Fluids* **30,** 047105. https://doi.org/10.1063/1.5024595 (Apr. 2018).
17. Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K. & Kaushik, S. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics* **64,** 525–545 (2019).
18. Thuerey, N., Weißenow, K., Prantl, L. & Hu, X. Deep Learning Methods for Reynolds-Averaged Navier–Stokes Simulations of Airfoil Flows. *AIAA Journal* **58,** 25–36 (2020).
19. Wu, H., Liu, X., An, W., Chen, S. & Lyu, H. A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils. *Computers and Fluids* **198,** 104393. ISSN: 0045-7930 (2020).
20. Sirovich, L. Turbulence and the dynamics of coherent structures. I. Coherent structures. *Quarterly of applied mathematics* **45,** 561–571 (1987).
21. Eckart, C. & Young, G. The approximation of one matrix by another of lower rank. *Psychometrika* **1,** 211–218 (1936).
22. Duchon, J. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique* **10,** 5–12 (1976).
23. Goodfellow, I. J., Bengio, Y. & Courville, A. *Deep Learning* http://www.deeplearningbook.org (MIT Press, Cambridge, MA, USA, 2016).
24. Soleimany, A. Deep Generative Models. *MIT 6.S191* **1,** 11–17 (2020).
25. Forrester, A. & Keane, A. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences* **45,** 50–79 (Apr. 2009).

26. Schwamborn, D., Gerhold, T. & Heinrich, R. The DLR TAU-code: recent applications in research and industry (2006).
27. Wu, Z. *et al.* A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* **32,** 4–24 (2020).
28. Bekemeyer, P. *et al. Data-Driven Aerodynamic Modeling Using the DLR Smarty Toolbox* in *AIAA Aviation 2022 Forum* (June 2022).
29. Paszke, A. *et al.* in *Advances in Neural Information Processing Systems 32* (eds Wallach, H. *et al.*) 8024–8035 (Curran Associates, Inc., 2019). `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.
30. Abadi, M. *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. `https://www.tensorflow.org/`.
31. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
32. Vaswani, A. *et al.* Attention is all you need. *Advances in neural information processing systems* **30** (2017).

7 APPENDIX

Table 4: Interpolation hyperparameters

Hyperparameter	Possible Values
interpolator	Linear, Square, Cubic, TPS
	Gaussian, Gaussian-Exponential, Cubic Spline
augmentation	None, Constant, Linear, Quadratic
scale	False, True
regularization	None, True
tune modus	None, Maximum-Likelihood (ML)

Table 5: Optimum interpolation hyperparameters

Hyperparameter	POD+I	Isomap+I	AE+I
interpolator	TPS	Linear	TPS
augmentation	Constant	None	Linear
scale	True	True	True
regularization	True	None	True

Table 6: MLP-Decoder hyperparameters

Hyperparameter	Range	Optimum
Learning rate	$[0.0001, 0.1]$	7.50×10^{-3}
Number of hidden layers L	$\{7, 8, 9, 10\}$	8
Downsampling multiplier r	$[0.8, 0.9]$	0.838

Table 7: MLP-Pointwise hyperparameters

Hyperparameter	Range	Optimum
Learning rate	$[0.000001, 0.01]$	5.62×10^{-4}
Number of hidden layers L	$\{7, 8, 9, 10\}$	10
Downsampling multiplier r	$[0.9, 1.0]$	0.906

Table 8: AE hyperparameters

Hyperparameter	Range	Optimum
Learning rate	$[0.00001, 0.01]$	1.64×10^{-4}
Number of hidden layers L	$\{2, 3, 4\}$	2
Downsampling multiplier r	$[0.7, 0.8]$	0.700

Table 9: Layer Dimensions

Model	Dimensions
MLP-Decoder	$[2, 75, 89, 106, 126, 150, 179, 214, 256, 110169]$
MLP-Pointwise	$[5, 256, 232, 210, 190, 172, 156, 141, 128, 116, 105, 1]$
AE+I	$[110169, 256, 179, 64, 179, 256, 110169]$