

Original article

Predicting onflow parameters using transfer learning for domain and task adaptation

Emre Yilmaz ^{*}, Philipp Bekemeyer 

German Aerospace Center, Institute of Aerodynamics and Flow Technology, Center for Computer Applications in AeroSpace Science and Engineering, Lilienthalplatz 7, Braunschweig, 38100, Niedersachsen, Germany

ARTICLE INFO

Keywords:

Transfer learning
Neural networks
Onflow parameter prediction
Nonlinear systems
Domain adaptation
Task adaptation

ABSTRACT

Determining onflow parameters is crucial from the perspectives of wind tunnel testing and regular flight and wind turbine operations. These parameters have traditionally been predicted via direct measurements which might lead to challenges in case of sensor faults. Alternatively, a data-driven prediction model based on surface pressure data can be used to determine these parameters, which requires complex system representations including nonlinearities. It is essential that such predictors achieve close to real-time learning as dictated by practical applications such as monitoring wind tunnel operations or learning the variations in aerodynamic performance of aerospace and wind energy systems. To overcome the challenges caused by changes in the data distribution as well as in adapting to a new prediction task, we propose a transfer learning methodology to predict the onflow parameters, specifically angle of attack, α , and onflow speed, V_∞ . It requires first training a convolutional neural network (ConvNet) model offline for the core prediction task, then freezing the weights of this model except the selected layers preceding the output node, and finally executing transfer learning by retraining these layers. A demonstration of this approach is provided using steady CFD analysis data for an airfoil for i) domain adaptation where transfer learning is performed with data from a target domain having different data distribution than the source domain and ii) task adaptation where the prediction task is changed. Further exploration on the influence of noisy data, performance on an extended domain, and trade studies varying sampling sizes and architectures are provided. Results successfully demonstrate the potential of the approach for adaptation to changing data distribution, domain extension, and task update while the application for noisy data is concluded to be not as effective. For the former adaptation cases, we have achieved test set mean absolute error values below 10^0 and as good as 10^{-1} using deep ConvNet architectures.

1. Introduction

Estimating critical onflow parameters such as angle of attack, α , and onflow speed, V_∞ , with high accuracy is essential for reliably assessing model characteristics in wind tunnel tests and ensuring safe and efficient flight and wind turbine operations [1–4]. These parameters have traditionally been determined by either direct measurements or statistical inference based on sensor data [1,5–8]. The conventional measurement approaches for angle of attack include inertial and optical techniques [3,9–12] as described in [13]. Additionally, α is estimated by model-based [8] or data-driven [14] inference schemes using the received state data via instruments such as Inertial Navigation system (INS) [5]. Similarly, flow velocity is typically determined using pressure tubes, anemometry, probes, and PIV as well as optical methods such as the Schlieren technique [1,15–18]. Furthermore, neural networks have

also been employed as a data-driven technique to estimate the components of the flow velocity vector based on surface pressure data [19–21].

Aircraft manufacturers rely on redundant systems for online monitoring of such critical parameters and leverage consistency tests and voting mechanisms [22]. Examples of the use of redundant systems in modern fly-by-wire aircraft include signal consolidation and monitoring which are executed using multiple α sensors, generally as many as three, before passing the data to the flight control system [23,24]. As an alternative, the incorporation of data-driven estimation models based on state information into voting systems has been proposed to address the risk of having two faulty sensors in agreement [25]. Inspired by these works, we further investigate the potential of data-driven techniques, specifically neural networks, for predicting onflow parameters based on surface pressure data.

^{*} Corresponding author.

E-mail address: emre.yilmaz@dlr.de (E. Yilmaz).

<https://doi.org/10.1016/j.ast.2025.111161>

Received 23 May 2025; Received in revised form 10 October 2025; Accepted 25 October 2025

Available online 2 November 2025

1270-9638/© 2025 The Authors.

Published by Elsevier Masson SAS. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Nomenclature

α	Angle of attack [deg or rad]
D_e	Extended data domain
D_i	Initial data domain
D_R	Data domain created using RANS + RSM
D_S	Data domain generated using RANS + SA
D_s	Source domain
D_t	Target domain
P	Surface pressure [Pa]
V_∞	Onflow speed [m/s]
b	Bias
CFD	Computational fluid dynamics
ConvNet	Convolutional neural network
DoE	Design of experiments
FCNN	Fully connected neural network
INS	Inertial navigation system
MAE	Mean absolute error
MSE	Mean squared error
NN	Neural network
n_{in}	Input dimension for FCNN
n_s	Number of surface data points
n_d	Dataset size
N_{OL}	NN trained during the offline learning phase
N_{TL}	NN trained during the transfer learning phase
OL	Offline learning
PDE	Partial differential equations
RANS	Reynolds-averaged Navier Stokes
RSM	Reynolds stress model
SA	Spalart-Allmaras one-equation turbulence model
\mathcal{T}_α	Task of predicting α
\mathcal{T}_{V_∞}	Task of predicting V_∞
TL	Transfer learning
t_{OL}	Training time for offline learning
t_{TL}	Training time for transfer learning
θ	Weights
θ_f	Frozen weights
θ_u	Unfrozen weights
μ	Mean
σ	Standard deviation

For applications as in this work, the systems of concern are highly complex and include nonlinear effects, and the system behavior is conveniently modeled using partial differential equations (PDEs) [26]. For fluid dynamics related problems, long-established computational techniques and tools exist to solve PDEs [27]. For extensive applications including nonlinear phenomena such as wave propagation, transformation techniques and symbolic programming have been considered to solve the corresponding PDEs [28–30]. Fundamental machine learning methodologies have also been employed for understanding, modeling, optimizing, and controlling fluid flows [26,31].

Recent trends in data-driven modeling related to aerodynamic applications are vastly influenced by emerging deep learning techniques which achieved remarkable accomplishments in many fields such as decision making [32], natural language processing [33], and computational biology [34]. Neural networks with fully connected layers have been employed for predicting aerodynamic coefficients before the breakthroughs in deep learning concepts that occurred over the previous decade [35–37]. With the introduction of deep architectures, there has been an expanding use of these techniques promising data-driven models that can extract physical correlations from flow field data. These efforts contain flow field estimation [38–44], generative design and shape optimization [41,45–47], and turbulence modeling [48]. Motivated by these advancements, we explore deep convolutional neural net-

work (ConvNet)-based architectures to learn the mapping from sensor pressure readings to onflow condition.

The lifelong variations in system behavior or changes in the environment can also lead to further challenges when determining onflow parameters based on sensor data [49,50]. The environmental impacts present during operations or tests may alter the distribution of physical data and require the data-driven model to adapt to this new domain via further updates [51]. In this context, transfer learning, a technique effectively used in computer vision [52], has been applied to force identification problems during wind tunnel testing [53], predicting flow fields around airfoils [54–56], inverse airfoil design [56], performance change due to damage [57], aerodynamic load estimation [58], and anomaly detection in wind turbines [59]. In these studies, knowledge transfer from the domains with abundant data to the domains with sparse data has been investigated [53,54,56,58,59]. With the transfer learning approach, the training optimization can be conducted at reduced computational cost and requires lower training times [60].

With the objective of overcoming challenges related to domain and task adaptation when predicting onflow parameters based on surface pressure data, a deep learning-based framework is created incorporating transfer learning. The specific onflow parameters to be predicted are selected as i) onflow speed, V_∞ , and ii) angle of attack, α . As a demonstration case, 2-D flow around an airfoil is considered, and the representative datasets are generated using a computational fluid dynamics (CFD) code. The learning architectures evaluated in the framework are neural networks with fully connected or convolutional layers.

For the synthetic model-free angle estimator proposed in [14], a nonlinear least squares formulation is solved using the Levenberg-Marquardt algorithm, which requires the linearization of the mapping from the inputs to α . Since the linearization can result in poor performance considering the existence of highly linear dynamical behavior and the inversion of the corresponding dynamical mapping as in our case, we select convolutional neural networks as the learning architecture in our framework similar to the situational awareness model in [21]. The V_∞ and α prediction problems in [21] are formulated as classification tasks requiring the discretization of the output space. For the same prediction problems, we assume a regression formulation without discretizing the output space in order to achieve better prediction accuracy.

The contributions of the proposed methodology can be summarized as follows:

1. A ConvNet based data-driven model to predict onflow parameters, specifically α and V_∞ , based on surface pressure data
2. A transfer learning framework that enables updating the network models with new data for domain and task adaptation
3. Demonstration cases for adaptation to changing data distribution, domain extension, and task change on the same domain as well as robustness to noise for 2-D flow around an airfoil

The remainder of this paper is organized as follows. First, we introduce the transfer learning approach to predicting the onflow parameters providing further theoretical details. Then, we explain the procedures of how to generate the representative aerodynamics data using a CFD solver for an airfoil geometry. Next, the offline learning is demonstrated for the task of predicting α and V_∞ with discussions about the selected parameters and the influence of the number of CFD runs, the number of points at which surface data is available, and the architecture depth. Finally, we present the transfer learning results for the scenarios of changing data distribution, domain extension, learning in a noisy domain, and task adaptation.

2. Methodology

In this section, we introduce how transfer learning can be employed to predict the onflow parameters, α and V_∞ , based on surface pressure data when either the underlying data distribution of the domain or the

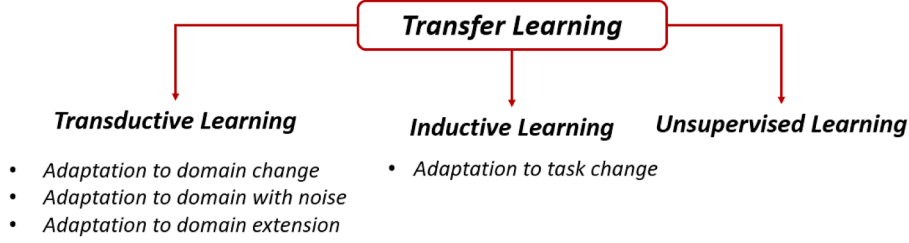


Fig. 1. Categorization of transfer learning.

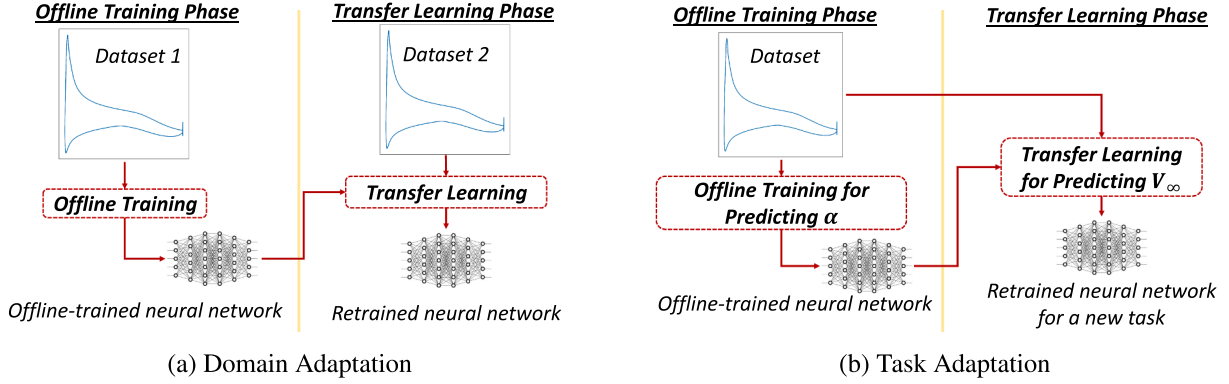


Fig. 2. Transfer learning framework.

type of task changes. This approach can pave the way to achieve online training with streaming data in wind tunnel experiments and during operations and thus enable predictions that can take the changes in the domain of operation into account. In addition, the knowledge obtained from the offline learning phase can be transferred to perform related tasks such as predicting a different onflow parameter than the one for which the source model was trained offline. Furthermore, we give a brief summary about the deep learning concepts such as convolutional neural networks (ConvNets) that are used to demonstrate the transfer learning framework.

Early literature related to transfer learning includes the discussions about concepts such as knowledge transfer from one task to another, life-long learning, and adaptation based on prior learning [61–63]. In a more recent survey [60], tasks are divided into the categories of inductive, transductive, and unsupervised transfer learning. Following the notation and categorization in [60], the types of transfer learning are presented with several examples in Fig. 1, and the formal definitions are provided as follows. Given a domain, \mathcal{D} , with a feature space, \mathcal{X} , and a marginal probability distribution, $P(X)$, where $X \in \mathcal{X}$, a task, τ , consists of learning a function that can make predictions on an output space, \mathcal{Y} . The objective of transfer learning is to improve the learning process for a target task, τ_t on a target domain, \mathcal{D}_t using the knowledge obtained executing the learning task, τ_s , on the source domain, \mathcal{D}_s . In transductive transfer learning, learning experience is transferred from a source domain, \mathcal{D}_s , to a target domain, \mathcal{D}_t , with different but similar characteristics for the same task. For inductive transfer learning, source task, τ_s , and target task, τ_t , are different but related to each other while source and target domains have the same data distribution. Although both domains and tasks differ from each other in the case of unsupervised learning, they are still sufficiently similar to each other. A significant advancement related to the utilization of transfer learning is the pretraining and task specific tuning of large language models (LLMs) such as BERT [64] and GPT-3 [33].

For the task of predicting onflow parameters, we propose a transfer learning framework for domain and task adaptation. The transfer learning framework used for domain adaptation in this paper is described in

Fig. 2a. In this case, for the same prediction task (i.e. either predicting α or V_∞), the neural networks, N_{OL} , trained offline using data from the source domain can be leveraged for the training optimization of another neural network, N_{TL} , on the target domain via transfer learning. Conceptually, the source and target data can be composed of CFD runs or data from wind tunnel experiments or real life operations. Moreover, the transfer learning approach can be used for task adaptation as described by Fig. 2b when transferring the knowledge learned for the task of predicting α to the task of predicting V_∞ . In this case, the domains of source and target have the same data distribution whereas the prediction task changes.

As the first step of transfer learning, we train a neural network offline for a given prediction task using available data which corresponds to executing the source task on the source domain. Then, another neural network with the same architecture is initialized using the weights of the pretrained neural network. The first step consists of training the weights, θ , of a neural network architecture, $f_{NN}(\cdot)$, given a loss function, \mathcal{L}_s for the source task and the datasets $(\mathbf{X}, \mathbf{y})_s$ representing the feature and output spaces of the source domain as shown in Eq. (1). In this equation, the trained weights that minimize the loss function for the source task are defined as θ_s . After selecting the layers to be frozen for the target task, θ_s can be separated into two components i) the weights to be updated for the target task, θ_u and ii) the weights that are frozen for the target task, θ_f . As the final step, the unfrozen weights can be updated for the target task using the data on the target domain as shown in Eq. (2).

$$\theta_s = \arg \min_{\theta} \mathcal{L}_s(\theta, (\mathbf{X}, \mathbf{y})_s, f_{NN}) \quad (1)$$

$$\theta_t = \arg \min_{\theta_u} \mathcal{L}_t(\theta_u, \theta_f, (\mathbf{X}, \mathbf{y})_t, f_{NN}) \quad (2)$$

In a ConvNet based architecture, the weights of the convolutional layers and several fully connected layers following it are generally frozen while the remaining layers preceding the output node are re-trained. This results from the assumption that the features extracted on source targets using convolutional operations are generic and can be reused for target tasks [52]. Furthermore, the ability of convolutional

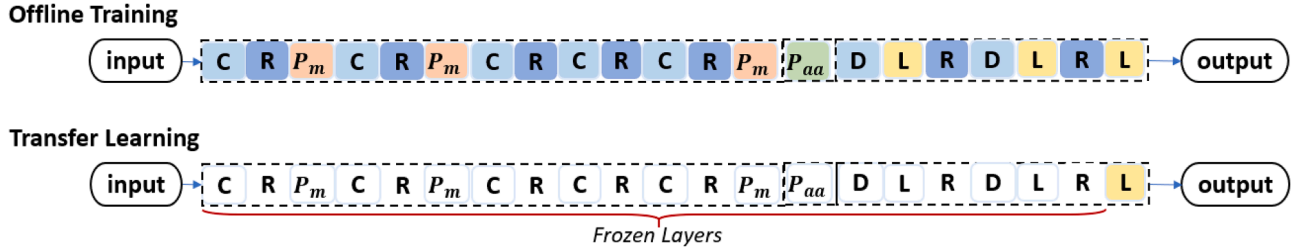


Fig. 3. ConvNet-D (C: convolutional, R: ReLU, P: pooling, m: max, aa: adaptive averaging, D: dropout, L: linear).

layers to generalize to different datasets are investigated, and it is concluded that more discriminative features are learned as the feature hierarchies become deeper [65]. As a result of the training session with the unfrozen weights, the knowledge extracted on the source domain for the source task can be transferred to execute the target task on the target domain.

At the core of our transfer learning framework, the main prediction task is executed via ConvNet based architectures. Originally inspired by the receptive fields in visual cognition systems, ConvNets are a class of neural networks with at least one layer that contains so-called “convolutional” filters to generate feature maps [66]. These filters are slid throughout inputs to apply convolution operations. ConvNets are particularly efficient in detecting local features of neighboring elements by applying a series of convolutional filters to the input images to extract various features from data.

ConvNets have resulted in extraordinary achievements in a vast variety of computer vision applications [67–69]. At the basis of this idea lies the assumption that local features in an input to the network (e.g. an array representing an input image) remain stationary, and therefore, ConvNets are effective in extracting these local features by executing computations on localized input subregions [66]. Additionally, the concept of weight sharing is exploited with ConvNets where the weights of each filter are shared throughout the spatial locations. Since the same weights are used for the entire input array, extracting local features invariant to translation is possible. In addition to the convolutional operation, a ConvNet-based architecture can include pooling and dropout operations, fully connected linear layers, batch normalization, and activation via function such as Rectified Linear Units (ReLU) [70].

The two ConvNet architectures selected for the learning tasks in this paper are shown in Figs. 3 and 4. They can be seen as representatives for a relatively deep and shallow network, respectively. The deeper architecture labeled as “ConvNet-D” is selected by converting the AlexNet architecture [71] implemented in the Torchvision models [72] into a one dimensional (1-D) form. This 1-D form allows inputting 1-D arrays to networks and then applying 1-D filters throughout these input arrays. It should be noted that 1-D convolution in PyTorch, f_{conv1D} , which is shown for the j^{th} element of the output in Eq. (3), is applied as cross-correlation [73]. In this equation, the valid cross-correlation operator is defined as \star whereas N_b , C_o , C_i , L , b , and h indicate the batch size, the number of output and input channels, the length of the 1-D array, bias, input to the convolution, respectively.

$$f_{\text{conv1D}}(N_b, C_o, j) = b(C_o, j) + \sum_{k=0}^{C_i-1} w(C_o, j, k) \star h(N_b, k) \quad (3)$$

The shallower architecture labeled as “ConvNet-S” is created by removing several of these layers. Figs. 3 and 4 describe how the weights of the layers of these network architectures are frozen during the transfer learning phase when training only the last layer. In these figures, convolutional layers, linear layers, rectified linear unit activations, max pooling operations, adaptive averaging pooling operations, and dropouts are represented by C, L, R, P_m, P_{aa}, and D, respectively. For comparison purposes, a fully connected neural network (FCNN) labeled as “FCNN” is also formed based on the linear layers of the shallow architecture.

3. Data generation

This section presents how the CFD analyses are executed to generate the representative data for aerodynamic performance of an airfoil. The generated data is assumed to represent the quasi-steady data obtained during i) an ordinary measurement task in a wind tunnel experiments, where tests are conducted based on a set of prefixed α and V_∞ values, and ii) steady flight or wind turbine operations within the operational envelope where the different combinations of V_∞ and α are held constant.

The steady CFD analyses are performed for the supercritical NLR 7301 airfoil [74] shown in Fig. 5 using the DLR-TAU Code [27] as flow solver. For all analyses, the Reynolds-averaged Navier-Stokes (RANS) equations are solved in conjunction with two different turbulence models, namely the Spalart Allmaras one-equation turbulence model (SA) [75] and the Reynolds stress model (RSM), SSG/LRR- $\ln\omega$, [76], in order to generate datasets with relevant but different distributions.

$$V_\infty \in [40, 70] \text{ m/s}$$

$$\alpha \in [-15, 17] \text{ deg} \quad (4)$$

To create the design of experiments (DoE) cases, α and V_∞ are chosen as factors within the boundaries shown in Eq. (4). The set of DoE cases are created using Halton’s method available in the DLR Surrogate Modeling for AeRo data Toolbox Python Package (SMARTy) [77]. The main advantages of the Halton’s method are the low discrepancy for lower dimensions and the flexibility to add further points to the generated datasets without changing the sampling characteristics due to its deterministic nature [78]. A subset of the generated DoE cases are shown in Fig. 6 by including the first 128 points for visualization purposes.

4. Results and discussion

In this section, we present the offline and transfer learning results for the problems of predicting α and V_∞ based on sparse surface pressure data. First, we provide an explanation of the steps for i) the parameter selection for the training optimization and demonstration cases, ii) how to preprocess the CFD data before the optimization phase, iii) how to split the data sets into training, validation, and test sets, and iv) the assessment of the performance of the trained models using evaluation metrics. Next, we discuss the prediction accuracy observed using the ConvNet-based architectures during the offline learning phase and provide several comparison cases in terms of the size of the entire dataset including training, validation, and test sets and the number of sampled surface measurement points. In this context, the dataset size corresponds to the number of samples selected from the entire CFD data set for the learning phases. Then, we discuss the transfer learning results regarding the demonstration cases for domain and task adaptation. The demonstration cases presented for domain adaptation include i) adaptation to a domain with relevant but different data distribution, ii) adaptation to a domain extension, and iii) adaptation to a noisy data domain. The feasibility of applying the transfer learning approach for task adaptation is also discussed on a demonstration problem where the source and target tasks are chosen as predicting α and V_∞ , respectively. For all demonstration cases, the variation of the results with the architecture type and

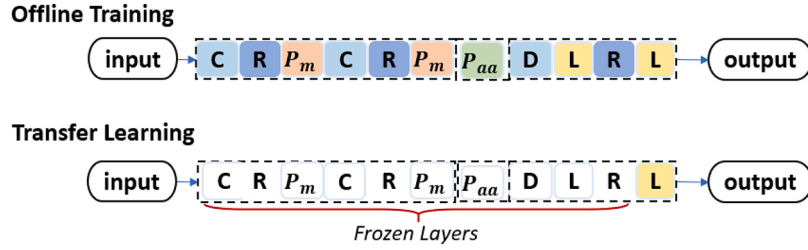


Fig. 4. ConvNet-S (C: convolutional, R: ReLU, P: pooling, m: max, aa: adaptive averaging, D: dropout, L: linear).

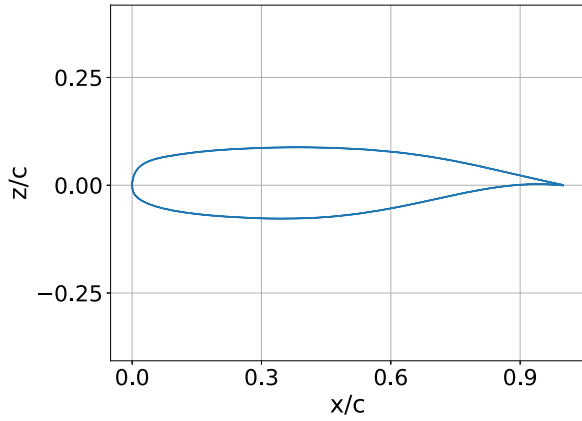


Fig. 5. NLR 7301.

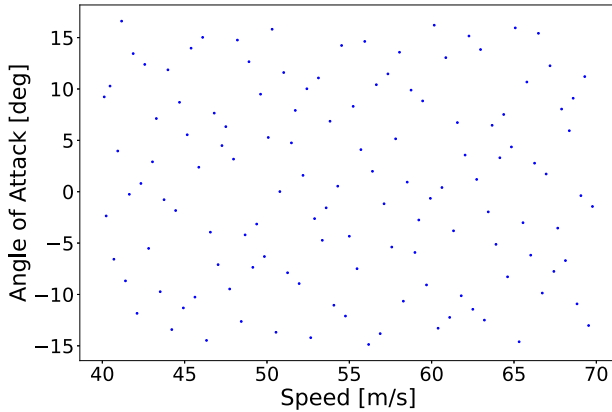


Fig. 6. Design of experiments for 128 CFD runs.

dataset size are included. Finally, an investigation of the required training time for the shallow and dense networks as well as the dataset sizes during the offline and transfer learning phases are provided.

4.1. Data preprocessing and splitting, parameter selection, and evaluation metrics

In this subsection, we explain the procedures for i) preprocessing and splitting the data and ii) selecting the parameters for the training optimization and the metrics that are used to evaluate the trained models. The pressure data on the airfoil surface obtained from the CFD analyses is cast into 1-D arrays and used as inputs to the neural network architectures. In this setting, convolution operations are executed with 1-D filters along these 1-D arrays. To investigate the variation of the results with the dimension of the input data, the surface data values shown in Table 1 are considered, and the input arrays are extracted from the CFD data accordingly. For each of these cases, surface pressure data points are selected equidistantly. Since the learning problem is essentially a

Table 1

Parameter sets for investigating prediction performance.

Parameter	Set
Surface data points	{38, 75, 150, 299, 597}
Number of CFD runs	{128, 256, 512, 1024}

regression problem, the output is a real-valued scalar corresponding to the onflow parameter of interest. Before the training optimization, inputs and outputs are normalized via min-max normalization such that their values remain between 0 and 1. In addition, the sensitivity of the training optimization to the size of the datasets used for both offline and transfer learning is investigated in terms of the test set prediction accuracy in the succeeding subsections. Therefore, the numbers of CFD runs expressed in Table 1 are used to create the datasets of different sizes. These datasets are then split into training, validation, and test sets and used for the training optimization of neural networks and the assessment of prediction accuracy.

As mentioned in Section 3, CFD runs are executed based on a set of DoE cases created by employing Halton sequences. Since this approach ensures low discrepancy as we add more points sequentially, it is agnostic to the number of samples in the set. Therefore, the generated DoE set is divided into training, validation, and test sets successively retaining the order of its elements. The splitting percentages of training, validation, and test sets are selected as 72%, 18%, and 10%, respectively.

Regarding the demonstration cases for adaptation to a domain extension, the datasets are first sorted in ascending order with respect to their output values and split from the median value into two subsets with equal numbers of elements. Then, these two subsets are sorted again based on the indices of their elements in the entire dataset before the first sorting operation. The resorting operation is executed to restore the Halton's sequence due to the reasons mentioned above. Next, the aforementioned procedure for creating training, validation, and test sets are followed for both subsets. Finally, offline training is performed on one half of the dataset, and transfer learning is employed by merging the training, validation, and test sets of both halves. For the remaining domain adaptation cases, the procedure for offline training is repeated with the new dataset.

We also investigate the influence of retraining only the last layer or the last two layers, which are fully-connected and linear, during the transfer learning phase. In case of one layer, we retrain 0.086% of the entire architecture for the shallow network and 0.017% for the dense one. Alternatively, if we choose to retrain two layers for both architectures, then the percentage of unfrozen weights are 98.697% for ConvNet-S and 70.359% for ConvNet-D. For these ConvNet architectures introduced in Section 2, the sequences and sizes of layers and convolution and pooling parameters are included in Tables A.1 and A.2 in the Appendices, respectively. For comparison purposes, the results obtained employing only a fully connected neural network architecture labeled as FCNN are also provided for the offline training step. The parameters of the fully connected neural network are given in Table A.3.

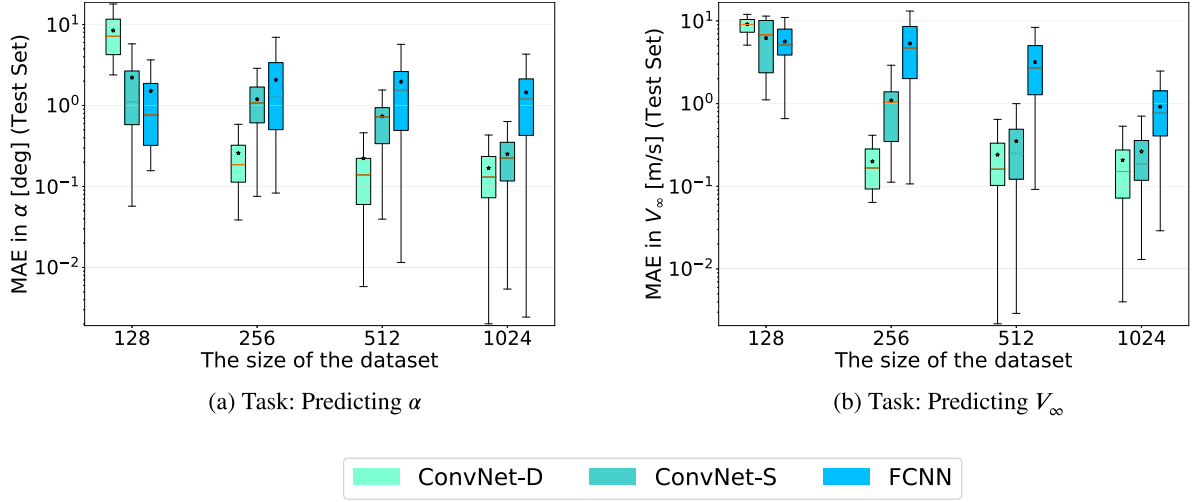


Fig. 7. The variation of the test set MAE values with the architecture type and dataset size for 75 surface data points.

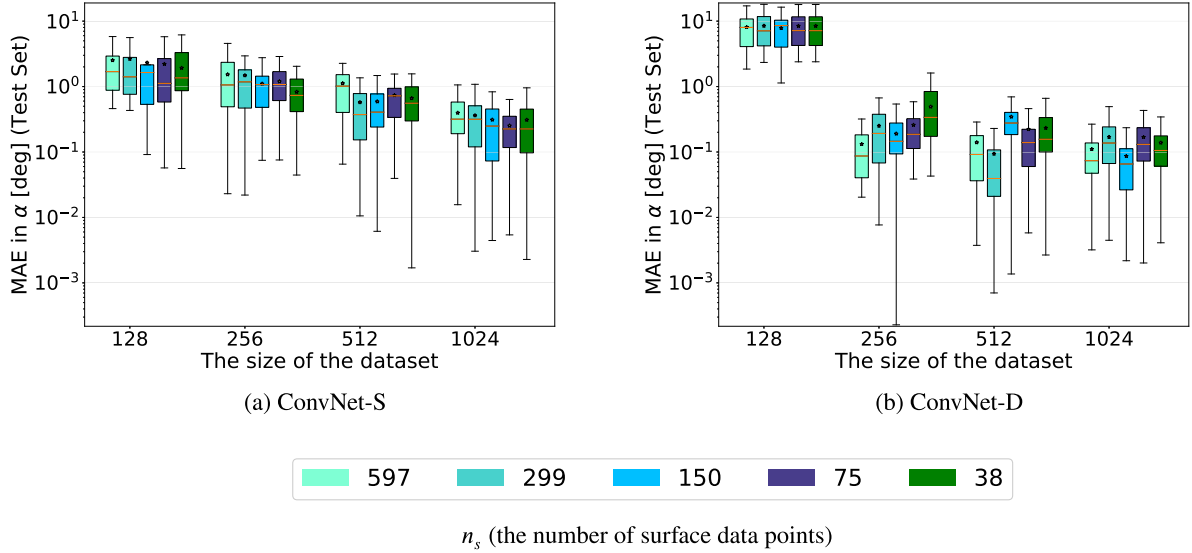


Fig. 8. The variation of the test set MAE values with the number of surface data points and dataset size (Task: Predicting α).

The tasks of predicting α and V_∞ are formulated as regression problems. The mean squared error (MSE) is selected as the loss function for the training optimization since it has a differentiable form, which is critical for gradient-based optimization and backpropagation. The contributions from outliers to MSE can result in misinterpretation of prediction errors. Alternatively, mean absolute error (MAE) is robust to outliers, and the metric for assessing the accuracy of predictions is, therefore, defined as mean absolute error (MAE). The relationships for these metrics are expressed in Eq. (5) where N_{test} is the size of the test set, y_i is the output of the i^{th} sample from the test set, which is used as the ground truth, and \hat{y}_i is the predicted value corresponding to the i^{th} sample.

$$\begin{aligned} \text{MAE} &= \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} |y_i - \hat{y}_i| \\ \text{MSE} &= \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (y_i - \hat{y}_i)^2 \end{aligned} \quad (5)$$

4.2. Predicting onflow parameters via ConvNet architectures

As the first demonstration case, the results obtained during the offline training phase are presented with discussions about how the pre-

diction accuracy varies with several factors. The prediction accuracy is measured by the test set MAEs as well as the variance, median, and quartiles of the test set absolute error values. Statistical values are derived by first computing the absolute errors for each entry in the test set, and then by analyzing the distribution of these errors. For the offline training phase, we investigate the influence of the architecture type, the numbers of points sampled on the airfoil surface, and the dataset size used for learning on the prediction accuracy. As explained above, the dataset size is defined as the sum of the sizes of the training, validation, and test sets for each case.

We first consider the variations of the test set MAE values with architecture type and dataset size, which are shown in Fig. 7a and b for the tasks of predicting α and V_∞ , respectively. For these cases, we fix the number of sampled surface data points to 75. It should be noted that the error values in the figures throughout the manuscript are shown in log scale. In comparison with the fully connected architecture, the ConvNet architectures achieve lower MAE values for both prediction tasks except for the minimum dataset size. Considering the practical applications, the MAE values for the minimum dataset size are found to be unacceptably high for all architectures. As the dataset size increases, the prediction accuracy achieved by the fully connected neural network is much worse than the ConvNet based architectures, and the correspond-

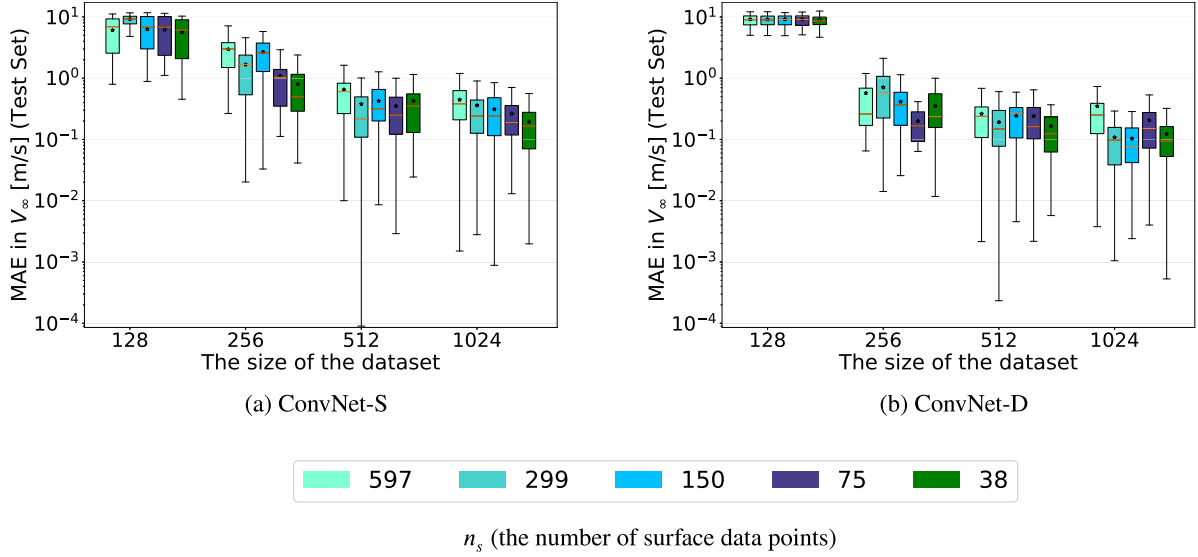


Fig. 9. The variation of the test set MAE values with the numbers of surface data points and dataset size (Task: Predicting V_∞).

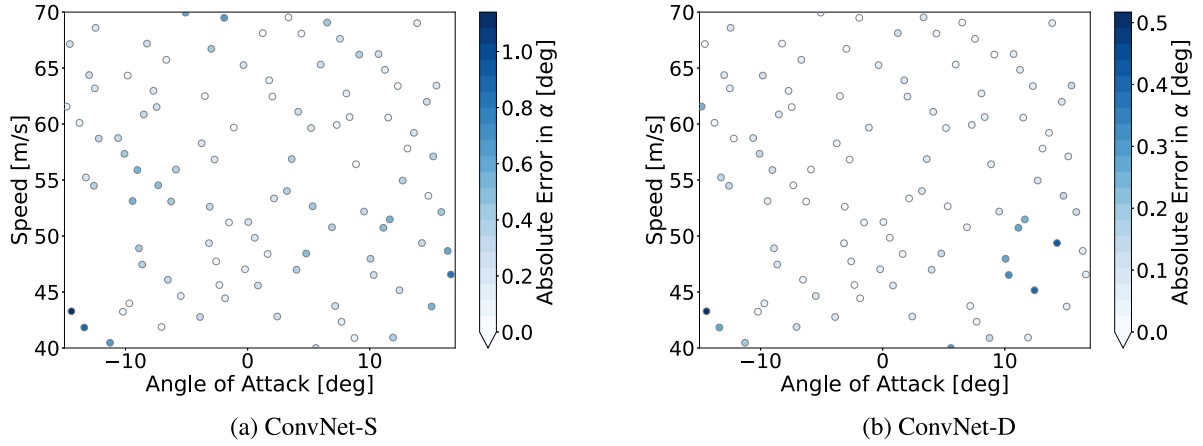


Fig. 10. Test set absolute errors for offline learning and the task of predicting α ($n_s = 75$, $n_d = 1024$).

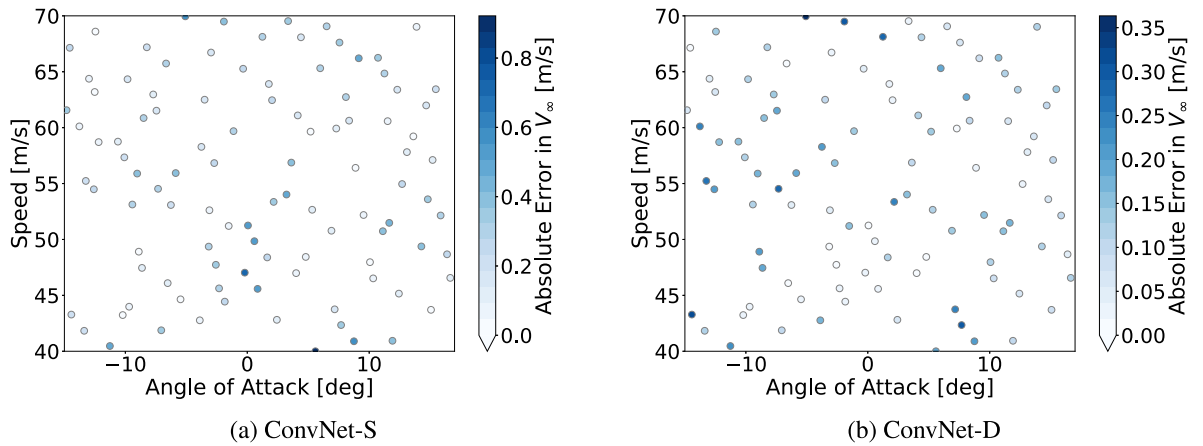


Fig. 11. Test set absolute errors for offline learning and the task of predicting V_∞ ($n_s = 75$, $n_d = 1024$).

ing MAE values cannot be reduced below 10^0 for both prediction tasks. Using the deep architecture, test set MAE values can be reduced to the level of 10^{-1} with a dataset size of 256 and do not improve significantly for the dataset sizes of 512 and 1024. For the shallow architecture, we observe more of a gradual trend where the MAE values decrease with the increase in dataset sizes and can approach the prediction accuracy of

the dense architecture when dataset size is set to 1024. Based on these observations, the results for the remaining cases are only provided using the ConvNet based architectures.

Next, we investigate the variation of the test set MAEs with the number of surface data points, where the pressure data is extracted, as well as dataset sizes. The results obtained for the problems of predicting α

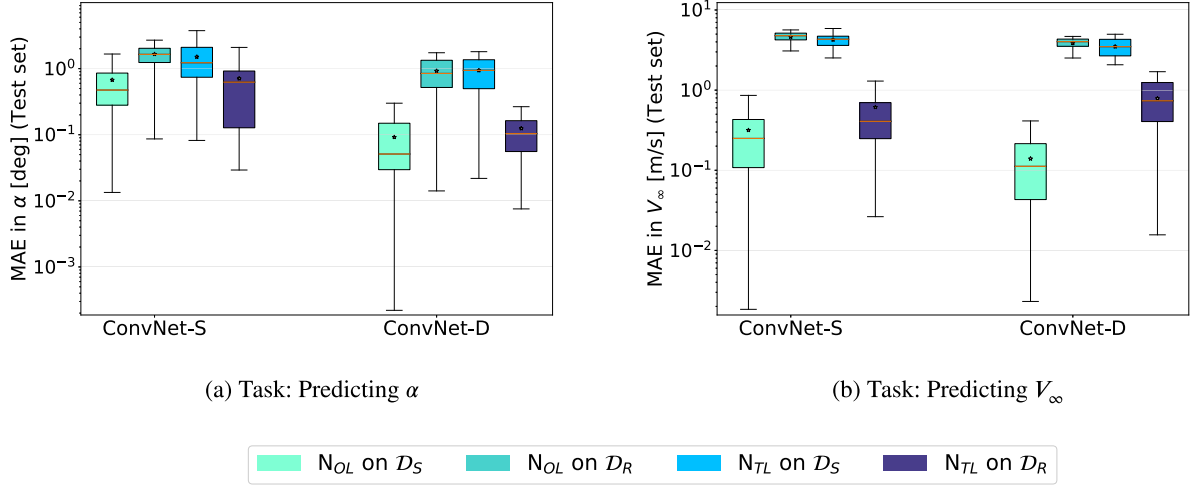


Fig. 12. Test set MAE values for transfer learning between the datasets generated using different turbulence models ($n_s = 75$, $n_d = 1024$).

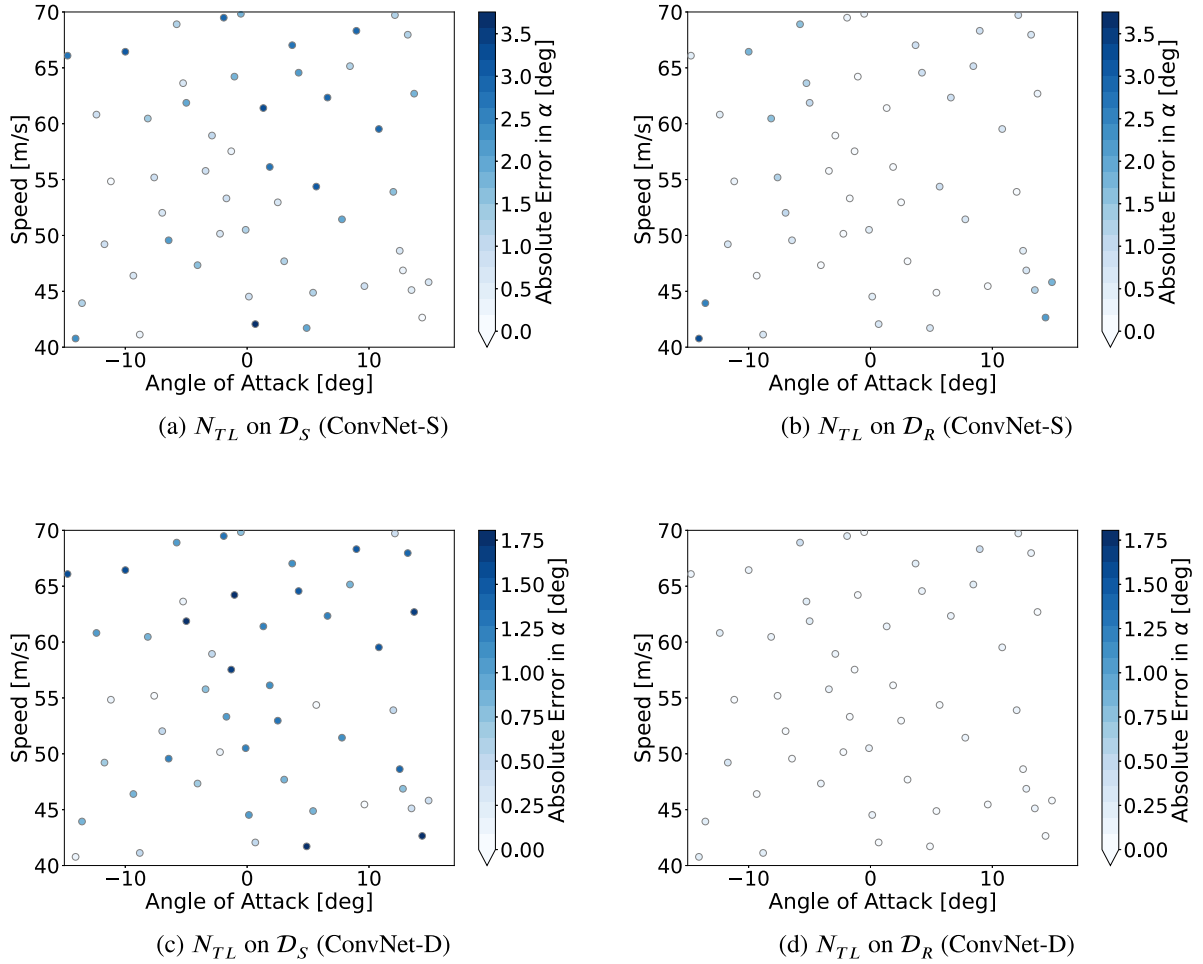


Fig. 13. Test set absolute errors obtained with N_{TL} for transfer learning between the datasets generated using different turbulence models and the task of predicting α ($n_s = 75$, $n_d = 1024$).

and V_∞ are shown in Figs. 8 and 9, respectively. The specific architecture is indicated under each subfigure. In this subsection, each bar plot for a fixed data set size indicates the number of surface data points as described by the corresponding legends. The surface pressure data is first extracted from the CFD data based on the meshed locations on the airfoil surface. Preserving the neighboring locations around the airfoil surface, an input array that contains the full surface information is

created. Then, for each downsampling case, a new input array is created by skipping a predetermined number of elements as points are selected equidistantly.

The observations from Figs. 8 and 9 are summarized as follows. In agreement with the previous observations, the MAE results attained for the smallest dataset size indicate that the number of samples are not adequate to learn the mapping ensuring the level of desired accuracy.

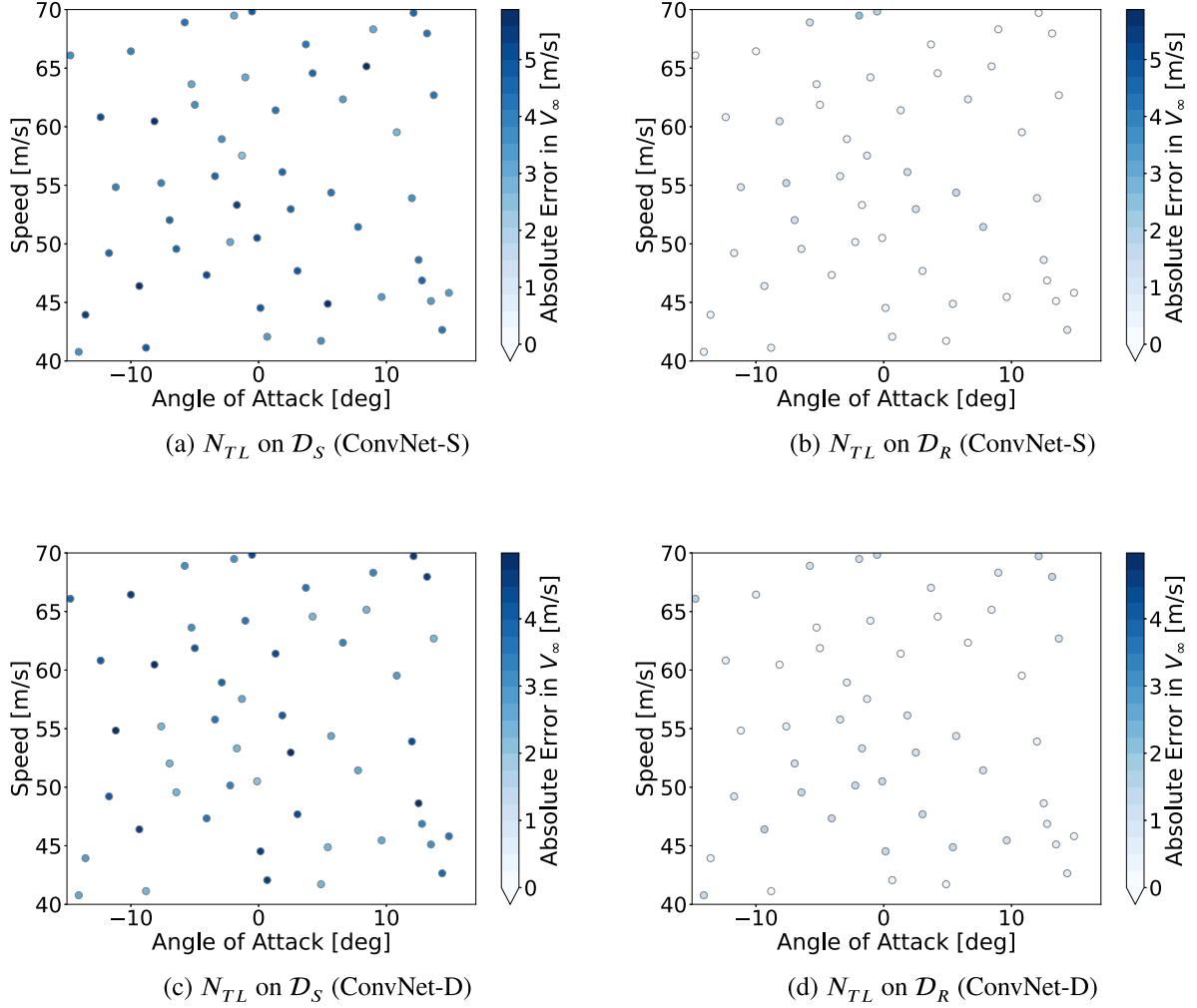


Fig. 14. Test set absolute errors obtained with N_{TL} for transfer learning between the datasets generated using different turbulence models and the task of predicting V_∞ ($n_s = 75$, $n_d = 1024$).

As the dataset size increases for both prediction cases, the MAE values have a tendency to decrease as expected. This decrease is much greater for the dense architecture whereas it follows a more gradual trend for the shallow architecture. When the variation of the MAE values with the number of sampled surface data points is inspected, we observe that there is no general trend for different dataset sizes, tasks, and networks. For example, the MAE values decrease as more surface points are sampled for a dataset size of 256 when using the ConvNet-D architecture to predict α . In order to extract the angle of attack information from the surface pressure data, local changes can be regarded as important factors, and the trend for this case, where local information is provided to the network at a finer resolution, seems to support this claim. Contrary to this, when employing a shallow network architecture for both tasks with a dataset size of 1024, the MAE values increase with increasing surface data points. In this contradictory case and the remaining cases with no clear trend, the amount of information that is already provided or the chosen pooling operations might be playing a role.

We provide the absolute error values of individual samples in Figs. 10 and 11 for these parameter values in order to gain a more in-depth insight. For α , the greatest absolute error values are obtained close to the lower limits of speed and angle of attack values or at higher angle of attack values below 55 m/s where separated flow is present. For speed, we observe a more random distribution of the absolute error values. For both prediction tasks, the maximum absolute error value obtained using

the ConvNet-S architecture is greater compared to ConvNet-D. Based on the discussed results, a dataset size of 1024 is concluded as sufficient to learn the mappings with prediction accuracy approximately resulting in an MAE value around 10^{-1} independent of the architecture. Therefore, we set surface data points, n_s , to 75 and the dataset size, n_d , to 1024 for the demonstration cases in the remaining parts of the manuscript.

4.3. Adaptation to the changes in data distribution

The first domain adaptation case explored in this paper is transferring the learned knowledge from one data domain to another which has a different data distribution. An important prerequisite for successfully employing transfer learning for this case is that the data distributions are related. As explained earlier in Section 3, the TAU flow solver is employed to compute flow solutions around the airfoil geometry based on the RANS equations. Additionally, in order to generate the datasets with different characteristics in this case, two different turbulence models, namely the Spalart-Allmaras one equation model and a Reynolds stress model, SSG/LRR- $\ln\omega$, are selected. The source domain, D_S , on which the offline learning is executed, is generated based on the RANS equations and the SA turbulence model. The target domain, D_R , to which the learning experience on D_S is transferred, is generated based on the RANS equations and using the Reynolds stress model, SSG/LRR- $\ln\omega$, for turbulence modeling. This results in two different data distributions within the same bounds which are nearly identical

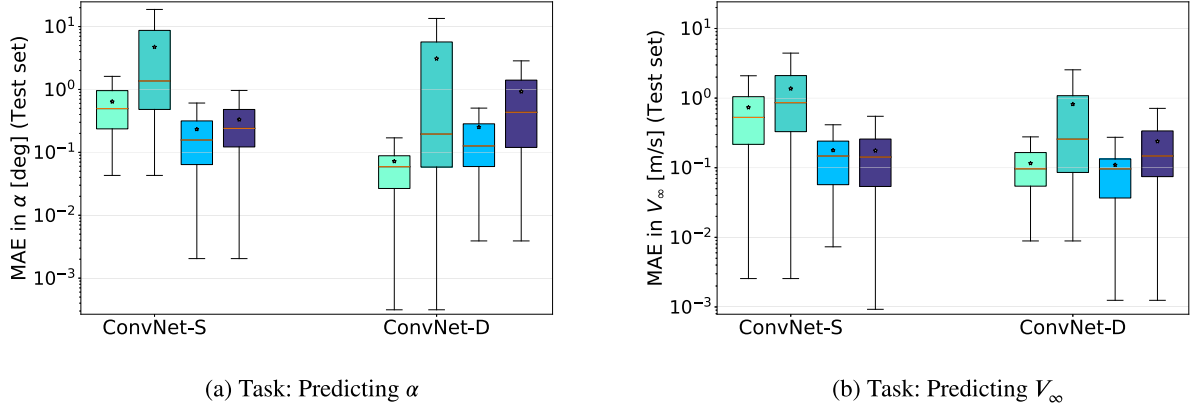


Fig. 15. Test set MAE values for transfer learning beyond the boundaries of the source domain ($n_s = 75$, $n_d = 1024$).

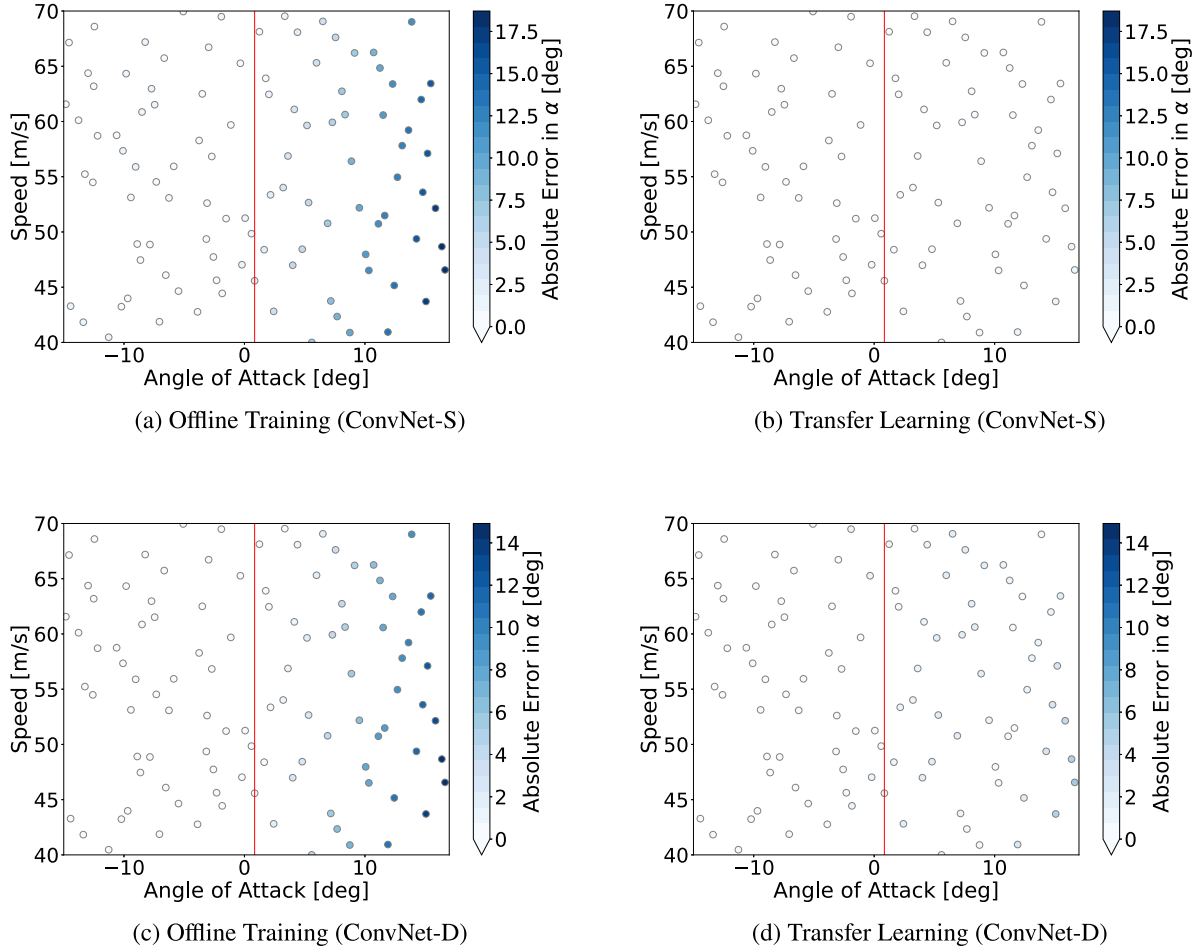


Fig. 16. Test set absolute errors for transfer learning beyond the boundaries of the source domain for the task of predicting α ($n_s = 75$, $n_d = 1024$).

in some parts of the domain including the cases at low angles of attack without separation. More severe differences exist in some other parts of the domain (e.g., larger angles of attack causing flow separation). A similar problem setup is most likely present when employing a pre-trained model during operations whether it be wind tunnel or flight testing.

We define the representations for the neural network architectures trained during the offline and transfer learning phases as N_{OL} and N_{TL} , respectively. For the remaining sections of the manuscript, these definitions are used consistently. As the first step in the transfer learning phase, the weights of N_{TL} are initialized with the weights of N_{OL} . Then, the weights of the selected frozen layers are kept fixed, while the re-

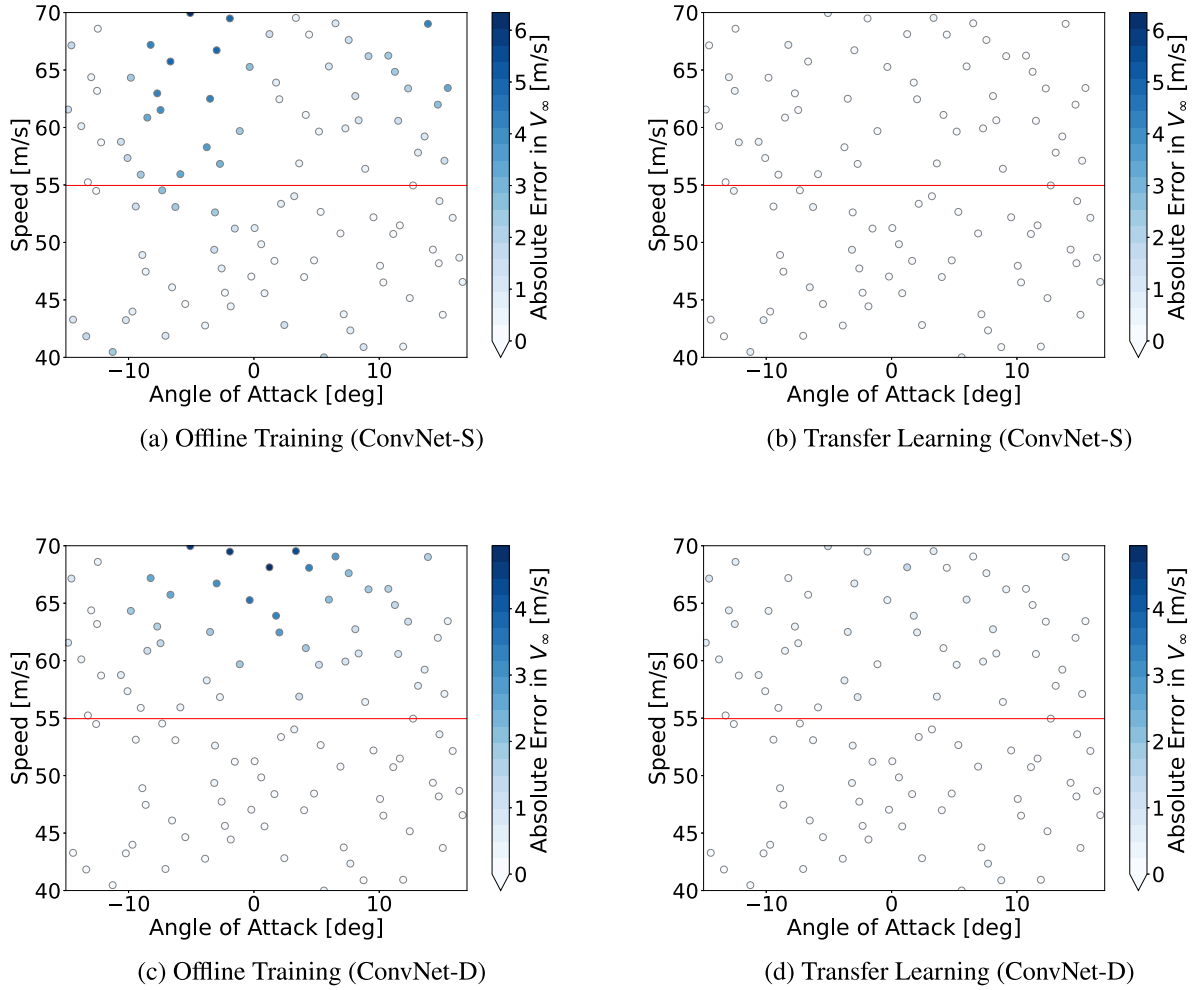


Fig. 17. Test set absolute errors for transfer learning beyond the boundaries of the source domain for the task of predicting V_∞ ($n_s = 75$, $n_d = 1024$).

maining weights are trained using the data from the target domain, D_R as explained in more detail in Section 2. For the demonstration case presented in this subsection, the variations of the test set MAE values with network architectures are shown in Fig. 12. In these figures, the legend labels, “ N_{OL} on D_S ” and “ N_{OL} on D_R ”, correspond to the MAE results achieved by the offline trained network, N_{OL} , on the test sets belonging to the source domain, D_S , and target domain, D_R , respectively. Similarly, the labels, “ N_{TL} on D_S ” and “ N_{TL} on D_R ”, indicate the MAE results achieved by N_{TL} on the test sets belonging to the aforementioned domains, D_S and D_R . To investigate the influence of architecture type on individual samples, the test set absolute error values obtained with N_{TL} are provided in Fig. 13 for α and in Fig. 14 for V_∞ for the results presented in Fig. 12. In addition, the test set absolute error values obtained with N_{OL} for this case are provided in Figs. A.1 and A.2 in the Appendices.

The test set MAE values obtained with N_{OL} and N_{TL} are higher for the domains, D_R and D_S , respectively, on which they are not trained to predict, compared to the results obtained on their intended domains of prediction. The differences between the corresponding error values can increase by as much as one order of magnitude. Besides, it is not possible to make accurate predictions on the source domain using the networks retrained on the target domain, and there is no specific pattern for the individual test set samples as shown in Figs. 13 and 14. For the individual absolute error values obtained on the target domain using the offline trained networks, the predictions are not accurate for the task of predicting V_∞ as shown in Figs. 14 and A.2. For the prediction task regarding α , as shown in Fig. A.1, N_{OL} can still achieve a desirable

level of accuracy and deviates from the ground truth for the higher α values of the domain.

These observations imply that the domains are sufficiently different to investigate the influence of transfer learning. In general, we expect that flow field solutions obtained using different turbulence methods, as demonstrated here, result in similar qualitative aerodynamic behavior as well as shifts and deviation of performance variables over the investigated angle of attack and speed ranges. For example, flow separation occurs as a nonlinear phenomenon in the flow field solutions regardless of the used turbulence model, and the angle of attack at which this phenomenon occurs can differ for different turbulence models. For such cases, transfer learning is a viable approach as demonstrated in this subsection.

4.4. Adaptation to a domain extension

In this section, the results for transfer learning beyond the boundaries of a data domain are presented. The initial domain used for the offline learning step, D_i , and the extended domain, D_e included for the transfer learning phase are created based on the output values of their elements. Further details about how to do the sorting and splitting operations for this case are explained in Section 4.1. Once the training, validation, and test sets on these domains are created, the offline training is first performed on D_i , and then the knowledge is transferred to the entire domain. From an engineering point of view, this corresponds to training an initial model based on numerical simulations that did not yield any results beyond a certain point, e.g. due to instabilities or lack-

ing convergence, while the actual model is exposed to such conditions during (physical) testing or operations.

For this demonstration case, the test set MAE results are shown in Fig. 15 for the tasks of predicting α and V_∞ . The corresponding network architecture is included in the horizontal axis of the subfigures. The test set MAEs obtained by using the offline trained network on the initial and extended domains are represented by " N_{OL} on D_i " and " N_{OL} on D_e ", respectively. Similarly, the test set results obtained using the network trained during the transfer learning phase are labeled as " N_{TL} on D_i " for the initial domain and " N_{TL} on D_e " for the extended domain. Besides, we present the individual absolute error values for each element in the test set on a grid of α and V_∞ values. The corresponding results for the tasks of predicting α and V_∞ are shown in Figs. 16 and 17, respectively.

Since the extended domain is not included during the offline training phase, the test set MAE results obtained on D_i by employing N_{OL} are lower than the results obtained on D_e . The offline training results in Figs. 16 and 17 also clearly indicate how the offline trained network models fail to predict for individual test set samples. Specifically, for the case of predicting α , the absolute error values increase for greater values of α on the extended domain, where flow separation and unsteady effects take over. While this pattern is common for both architectures when predicting α , the regions of the extended domain with greater test set absolute error values do not overlap for the task of predicting V_∞ . Overall, these outcomes suggest that the extrapolation performances of the offline learned models do not suffice for the prediction tasks. Although the initial domain is still included in the training and validation sets during the transfer learning phase, the capability of the network to predict on the initial domain is only improved for the shallow architecture. For the dense architecture, transfer learning results in either a similar performance as in the case of predicting V_∞ or degradation as for predicting α . For both architectures, the MAE results obtained on the extended domain improve significantly after transfer learning, and the MAE values are even reduced one order of magnitude for the shallow architecture. We assume that these observations stem from the percentage of weights that we enable to retrain during transfer learning, which is lower for the dense architecture, and hence, this architecture does not have as much capability as the shallow one to adjust to changes. On the contrary, the dense network is a better approximator than the shallow one when trained offline with data from the initial domain because it has more weights that can be tuned accordingly. Furthermore, changing the data distribution as done in the previous demonstration case yields a more common task for the models to adapt to during transfer learning while this seems to be more complicated for adapting to a domain extension.

4.5. Adaptation to a noisy data domain

As the final demonstration case for domain adaptation, we investigate the possibility of first learning without noise and then leveraging this offline trained model to learn on a noisy domain via transfer learning. For this purpose, a zero-mean, independent, and identically distributed Gaussian noise profile is assumed for each surface data point and added to the input data. This corresponds to receiving noisy readings from pressure sensors in an experiment or during an operational scenario. For each sample in the dataset, multiple noisy input data are considered to include the influence of the noise variance levels. For this specific study, 10 noisy inputs are generated for each sample, and therefore, these inputs correspond to the same output value. The network model is initially trained offline using the data from the noise-free domain labeled as D_i . Next, the weights of the selected frozen layers in the network model are fixed, and the remaining weights are retrained using the data from the noisy domain, which is represented by D_n , during the transfer learning phase. For the presented results, a set of variance values are considered. For the same prediction task and network architecture, each transfer learning phase with a different variance level is

initiated using the same offline trained model for the corresponding task and architecture.

The results for this case are shown in Figs. 18 and 19 for the tasks of predicting α and V_∞ , respectively. For both prediction tasks, the MAE values obtained with the noisy data after the transfer learning phase are much higher than the values corresponding to the offline learning phase in this demonstration case. Furthermore, these values are also higher than the values obtained during the transfer learning phases of the other demonstration cases. Since the weights of the unfrozen layers are updated during the transfer learning phase, the prediction accuracy obtained with N_{TL} on the noise-free domain degrades as well. The degradation increases for both shallow and dense networks as the noise variance levels increase. When using the ConvNet-D architecture during the transfer learning phase for both prediction tasks, the MAE values obtained on the noisy domain, D_n , are still higher than the values obtained on the noise-free domain, D_i . The corresponding values obtained on D_n for the shallow architecture are, however, lower than the MAE values obtained on D_i .

4.6. Adaptation to task change

In addition to domain adaptation, transfer learning can be used for adapting to task changes. A demonstration case is presented in this subsection where the source task is chosen as predicting α while the target task is set to predicting V_∞ . In this case, the source and target domains are the same and denoted as D . The network architecture is first trained on the domain, D , to predict α , and then transfer learning is applied on the same domain to predict V_∞ . Tasks are represented by τ with subscript referring to the predicted variable. For comparison purposes, we also include the results obtained using a network directly trained offline to predict V_∞ . The MAE values obtained for task adaptation are shown in Fig. 20 for the cases of training either one or two fully-connected layers before the output layer during the transfer learning phase. In this figure, the offline trained networks for the tasks of predicting α and V_∞ are represented by $N_{OL}(\tau_\alpha)$ and $N_{OL}(\tau_{V_\infty})$, respectively, whereas the network, which is trained to predict V_∞ based on $N_{OL}(\tau_\alpha)$ during the transfer learning step, is represented by $N_{TL}(\tau_\alpha \rightarrow \tau_{V_\infty})$.

Investigating the test set prediction accuracy for these networks, the MAE values achieved employing $N_{OL}(\tau_{V_\infty})$ can be deemed sufficient for a desired level of prediction accuracy in practical applications. It is also clear that the networks, $N_{OL}(\tau_\alpha)$, which are trained offline to predict α , fail to predict V_∞ . If one layer is retrained during the transfer learning phase, the prediction accuracy of $N_{TL}(\tau_\alpha \rightarrow \tau_{V_\infty})$ cannot be improved and performs similar to $N_{OL}(\tau_\alpha)$ in terms of the attained MAE values. When the weights of another layer are also included in the training optimization for transfer learning, the MAE values obtained employing $N_{TL}(\tau_\alpha \rightarrow \tau_{V_\infty})$ are reduced by one order of magnitude for the shallow architecture. This network can even achieve an error performance close to $N_{OL}(\tau_{V_\infty})$ for this case. By training two layers in this case, we retrain 98.697% of the weights in the shallow architecture and 70.359% of the weights in the dense architecture. However, the weights of the convolutional layers, which are used for feature extraction purposes, are fixed for both architectures. Therefore, transfer learning is not the same as nearly fully retraining the entire network.

Although the MAE values obtained employing $N_{TL}(\tau_\alpha \rightarrow \tau_{V_\infty})$ and the dense architecture are slightly improved, the results are still one order of magnitude higher than the values obtained employing $N_{OL}(\tau_{V_\infty})$. This might result from the percentage of the retrained weights, which is greater for the shallow network, and thus provides extra flexibility during the training optimization. Overall, the tasks of learning different onflow parameters based on sparse surface pressure data correspond to learning two different inverse maps. Transferring the learning experience regarding these inverse maps is a more complicated deed and requires that more layers be trained for better prediction accuracy.

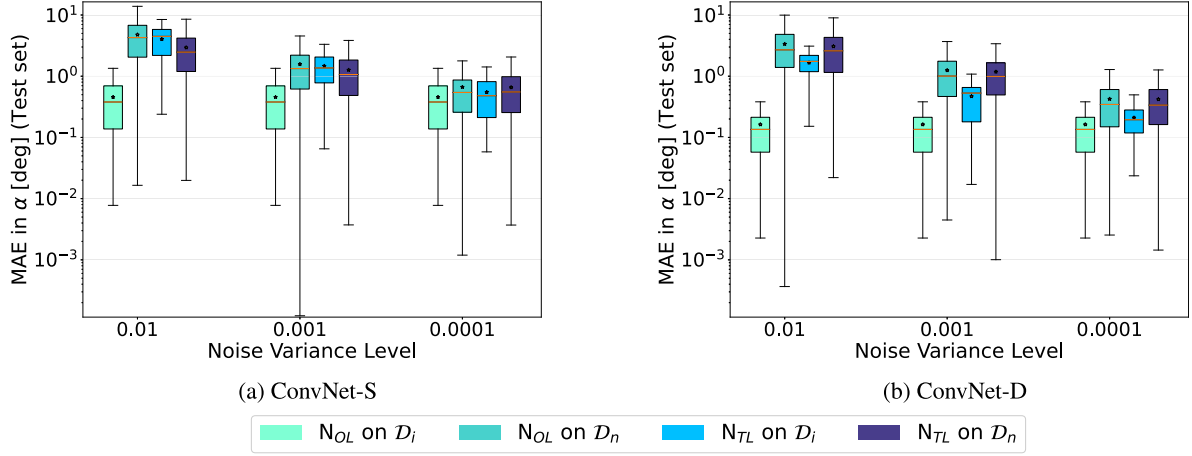


Fig. 18. The variation of the test set MAEs with noise levels for the task of predicting α ($n_s = 75$, $n_d = 1024$).

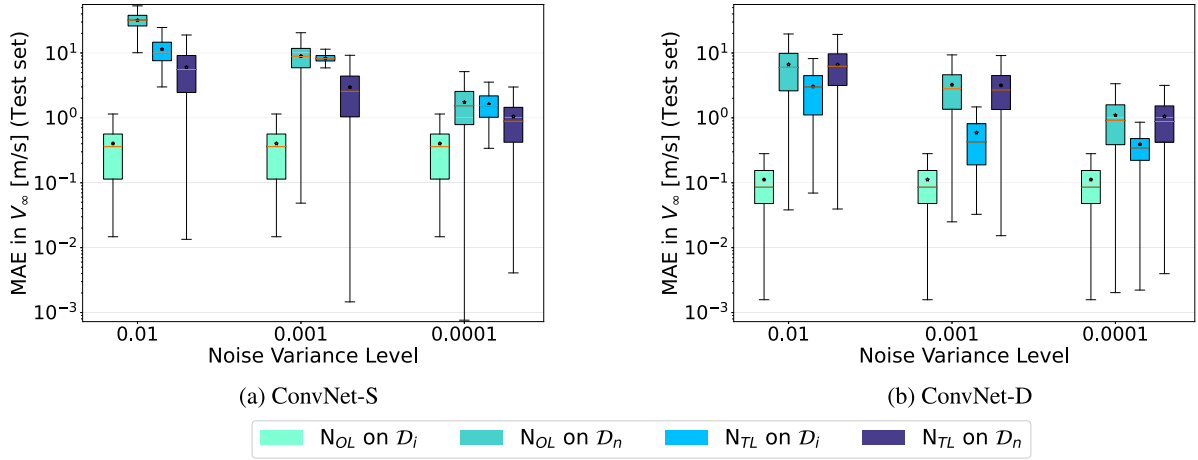


Fig. 19. The variation of the test set MAEs with noise levels for the task of predicting V_∞ ($n_s = 75$, $n_d = 1024$).

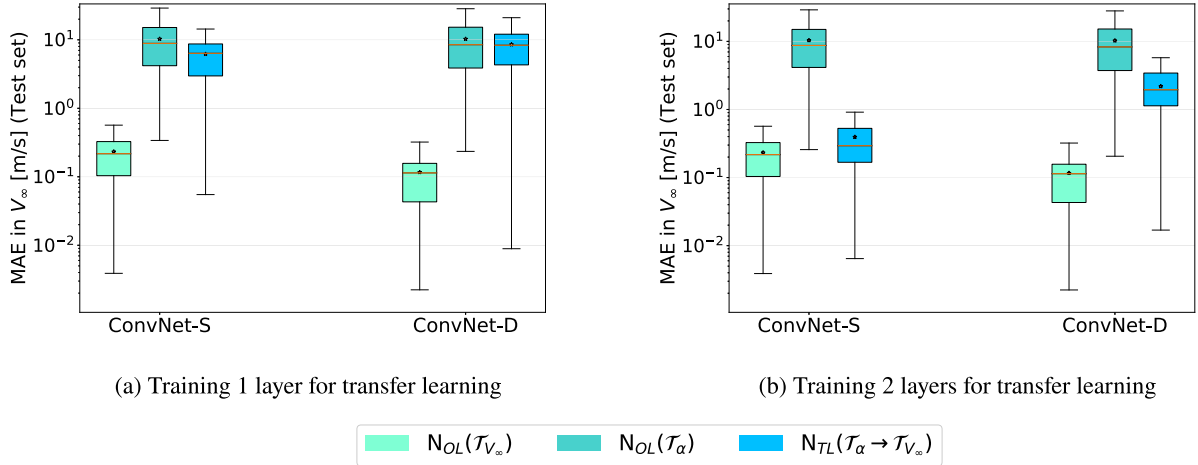


Fig. 20. Test set MAE values for the case of task adaptation to predicting V_∞ ($n_s = 75$, $n_d = 1024$).

4.7. Training time

For online applications such as wind tunnel experiments or wind turbine and flight operations, the required execution times are significant factors that determine the feasibility of real-time implementations. Considering the time limitations in this context, a real-time performance is

achievable by using the trained networks in this work for online prediction while it is not possible to train these neural networks in real time. Therefore, we also investigate whether the transfer learning approach can be leveraged to reduce the required training time with the ambition of approaching close to real-time performance while ensuring an acceptable level of prediction accuracy. For this purpose, we include

Table 2

Training time ($\mu \pm 1\sigma$) for the offline and transfer learning phases when adapting to changing data distributions in the domain.

Task: Predict α	ConvNet-S			ConvNet-D			
	Size	t_{OL} [s]	t_{TL}^{1L} [s]	t_{TL}^{2L} [s]	t_{OL} [s]	t_{TL}^{1L} [s]	t_{TL}^{2L} [s]
512		43.58 ± 6.65	11.03 ± 1.92	27.31 ± 1.04	142.75 ± 27.99	33.02 ± 4.07	23.30 ± 9.04
1024		58.76 ± 11.76	6.24 ± 1.66	21.61 ± 4.73	154.28 ± 42.35	17.62 ± 6.95	35.53 ± 12.40

Task: Predict V_∞	ConvNet-S			ConvNet-D			
	Size	t_{OL} [s]	t_{TL}^{1L} [s]	t_{TL}^{2L} [s]	t_{OL} [s]	t_{TL}^{1L} [s]	t_{TL}^{2L} [s]
512		63.21 ± 10.42	19.79 ± 4.11	14.93 ± 2.16	92.71 ± 22.14	53.60 ± 3.48	38.35 ± 21.01
1024		51.10 ± 7.50	8.40 ± 1.44	13.7 ± 2.88	113.03 ± 40.38	25.84 ± 5.67	55.19 ± 18.90

the training time results of the demonstration case for the adaptation to changing data distributions in this section. The training time results for offline and transfer learning phases, t_{OL} and t_{TL} , are included in Table 2 with additional information about tasks, dataset sizes, and architectures. Moreover, $1L$ and $2L$, which are the superscripts of t_{TL} , refer to retraining one layer and two layers during the transfer learning phase, respectively.

In general, the transfer learning methodology leads to reduced training times for both network architectures compared to a fully offline training. As the dataset size increases, the amount of reduction in training time is consistently greater for transfer learning by retraining one layer instead of retraining two layers. Nevertheless, it can be stated that the training times required by the transfer learning phases exceed the upper limits allowed for real-time learning.

5. Conclusions

In this manuscript, we have proposed a transfer learning framework for the problem of predicting the onflow parameters, namely angle of attack and speed, based on sparse surface pressure data. Since this problem fundamentally deals with inverting a mapping that describes highly nonlinear interactions in a complex system, a data-driven approach based on neural networks is used within the proposed framework. The data-driven model architecture that is used to learn the mapping from the surface pressure data to the onflow parameters, is selected based on convolutional neural networks due to their efficiency in learning on data while accounting for neighboring relationships. In this framework, a ConvNet based architecture is first trained on the source domain for the source prediction task offline. Then, the weights of the layers in this architecture except the final layer are kept fixed, and the remaining weights are retrained using the data from the target domain. In this context, transfer learning enables to make accurate predictions when there is a change in the flow characteristics or the prediction parameter of interest.

For the core task of predicting the onflow parameters, the ConvNet architectures lead to better prediction performance with increasing dataset size whereas fully connected neural networks are insufficient. The number of surface data points result in no consistent change in prediction accuracy. The dense network is concluded as better approximator for offline training because it has more weights that can be tuned accordingly. To investigate the benefits of using transfer learning, we have demonstrated domain and task adaptation cases. For the case of adaptation to the changes in data distribution, the prediction accuracy results achieved on the target domain using the network trained during the transfer learning phase are similar to those achieved on the source domain using the offline trained network. For the case of adaptation to a domain extension, the extrapolation performance of the offline trained network is unsatisfactory as expected. With the addition of more training data during the transfer learning phase, the prediction accuracy results on both the initial and extended domains are improved. The individual test set elements with higher absolute error values are observed to be close to the domain boundaries where nonlinear

phenomenon such as flow separation occurs. The transfer learning approach is ineffective in adapting to the domain with noise leading to much higher errors compared to the other demonstration cases. For the task and domain adaptation cases, the MAE values obtained with the retrained shallow ConvNet architecture are lower. This is assumed to result from a greater percentage of unfrozen weights that are retrained during transfer learning. Hence, the shallow architecture provides more flexibility for adaptation. Finally, although the transfer learning approach leads to reduced training times considering greater dataset sizes and denser architectures in particular, real-time learning has not been achieved.

Data-driven prediction models as in this manuscript can also pave the way for potential practical uses. They can serve as redundant backup systems for online monitoring to detect faulty behavior or as a part of the voting mechanisms in flight control system architectures. A future research objective is the investigation of the applicability of the proposed method to transfer the learned experience from CFD runs to wind tunnel experiments and to make accurate predictions online during the experiments. Other future directions include the exploration of alternative ways to improve the adaptation to the domains with noise and to reduce the training times required during the transfer learning phase. To improve robustness against noise, Bayesian neural networks, which are probabilistic graphical networks representing variables and their conditional dependencies, can be considered, since these methods address overfitting by modeling uncertainty in the weights. Furthermore, the tasks of learning different onflow parameters based on sparse surface pressure data correspond to learning two different inverse maps, and how much these inverse maps are related to each other is a question that requires further investigation.

CRedit authorship contribution statement

Emre Yılmaz: Writing – original draft, Visualization, Software, Methodology, Conceptualization; **Philipp Bekemeyer:** Writing – review & editing, Supervision, Resources, Methodology, Funding acquisition, Conceptualization.

Data availability

Data sharing can be discussed with the authors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge the scientific support and HPC resources provided by the German Aerospace Center (DLR). The HPC

system CARA is partially funded by “Saxon State Ministry for Economic Affairs, Labor and Transport” and “Federal Ministry for Economic Affairs and Climate Action”. The authors also gratefully acknowledge the funding support by the German Aerospace Center (DLR) through the SAFER2 project.

Appendix A.

A.1. Architecture parameters and other settings for transfer learning demonstration cases

In this subsection of the Appendices, we provide the parameters of the ConvNet-D, ConvNet-D, and FCNN architectures used for the demonstration cases. Specifically, Table A.1 includes the parameter information for the ConvNet-D architecture, Table A.2 includes the parameter

information for the ConvNet-S architecture, and Table A.3 includes the parameter information for the FCNN architecture.

The retrained layers for each transfer learning demonstration case and architecture are provided in Table A.4 with the corresponding figure numbers. For these demonstration cases, the corresponding data sizes and types are also shown in Tables A.5 and A.6, respectively.

A.2. Test set prediction accuracy of the offline trained neural networks for the case of adaptation to changing data distributions

The prediction accuracy results of the individual test set elements obtained using the offline trained neural networks are provided in Figs. A.1 and A.2 for the case of adaptation to changing data distributions and the tasks of predicting α and V_∞ , respectively.

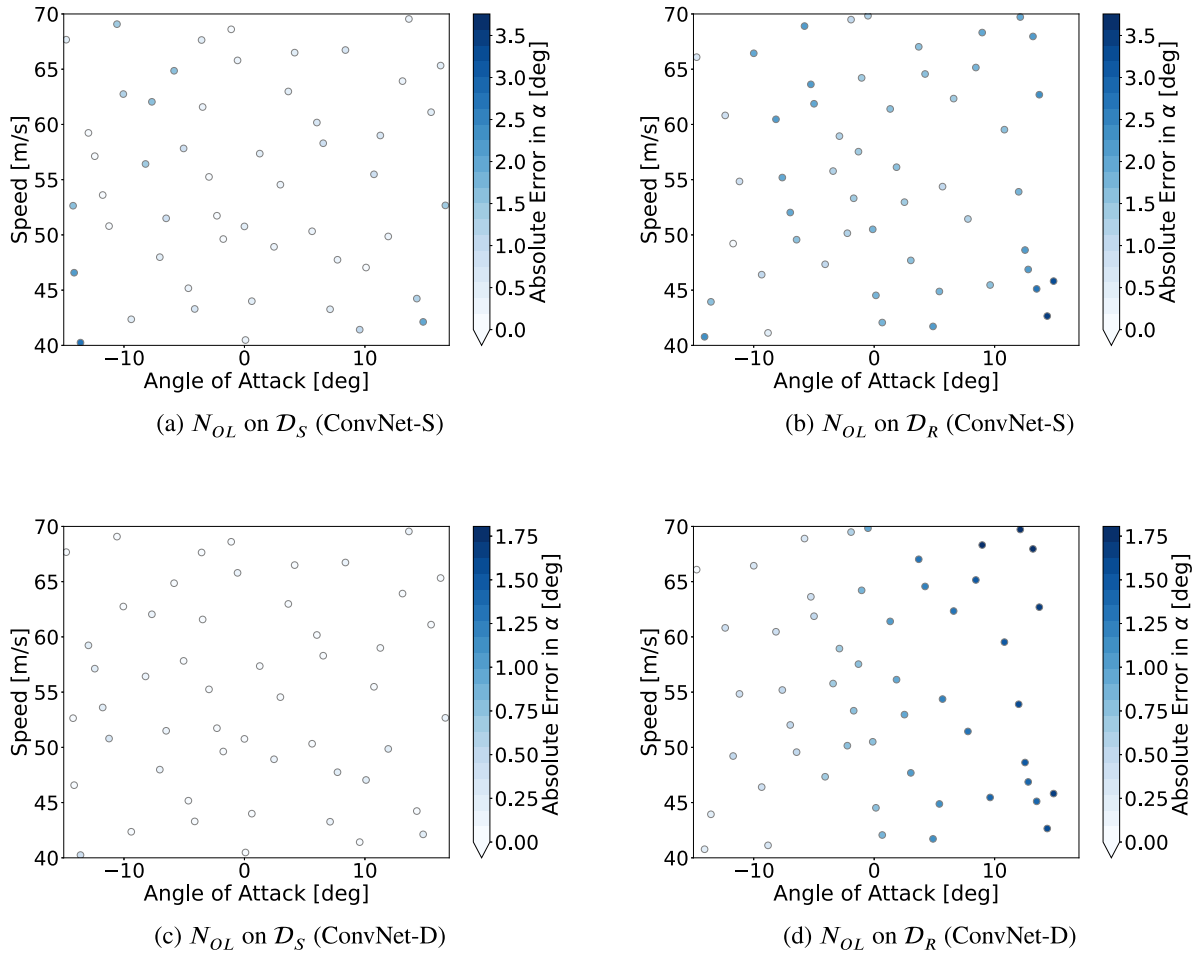


Fig. A.1. Test set absolute errors obtained with N_{OL} for transfer learning between the datasets generated using different turbulence models and the task of predicting α ($n_s = 75$, $n_d = 1024$).

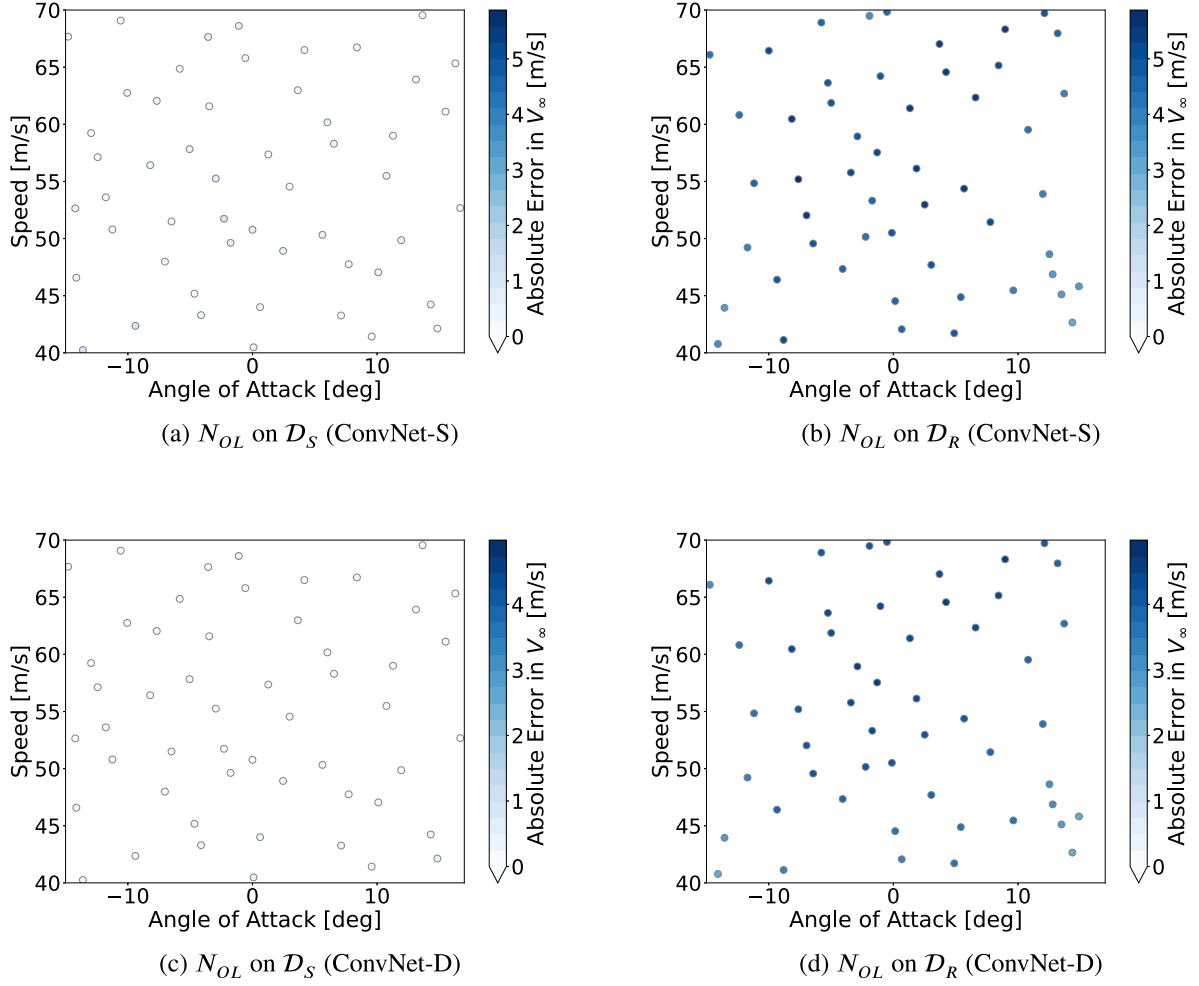


Fig. A.2. Test set absolute errors obtained with N_{OL} for transfer learning between the datasets generated using different turbulence models and the task of predicting V_∞ ($n_s = 75$, $n_d = 1024$).

Table A.1

The parameters of the ConvNet-D architecture.

Conv. Layer	size	Conv. Parameters	Linear Layer	size	Pooling Parameters
Conv1	(1, 64)	kernel size:11	Linear1	(1536, 4096)	Max Pooling
Conv2	(64, 192)	stride:4	Linear2	(4096, 4096)	kernel size:3
Conv3	(192, 384)	padding:2	Linear3	(4096, 1)	stride:2
Conv4	(384, 256)				
Conv4	(256, 256)				

Table A.2

The parameters of the ConvNet-S architecture.

Conv. Layer	size	Conv. Parameters	Linear Layer	size	Pooling Parameters
Conv1	(1, 64)	kernel sizes:(11, 5)	Linear1	(1152, 4096)	Max Pooling
Conv2	(64, 192)	stride:(4, -)	Linear2	(4096, 1)	kernel size:3
		padding:2			stride:2

Table A.3

The parameters of the FCNN architecture.

Linear Layer	size
Linear1	(n_{in} , 4096)
Linear2	(4096, 1)

Table A.4

The unfrozen layers during the transfer learning phase for the demonstration cases.

Section	Demonstration Case	Architecture	Unfrozen Layers	Figures
4.3	Adaptation to the changes in data distribution	ConvNet-S	Linear1, Linear2	12a,b, 13a,b, 14a,b, A.1a,b, A.2a,b
4.3	Adaptation to the changes in data distribution	ConvNet-D	Linear2, Linear3	12a,b, 13c, d, 14c,d, A.1c,d, A.2c,d
4.4	Adaptation to a domain extension	ConvNet-S	Linear1, Linear2	15a,b, 16a,b, 17a,b
4.4	Adaptation to a domain extension	ConvNet-D	Linear2, Linear3	15a,b, 16c,d, 17c,d
4.5	Adaptation to a noisy data domain	ConvNet-S	Linear1, Linear2	18a, 19a
4.5	Adaptation to a noisy data domain	ConvNet-D	Linear2, Linear3	18b, 19b
4.6	Task adaptation	ConvNet-S	Linear2	20a
4.6	Task adaptation	ConvNet-D	Linear3	20a
4.6	Task adaptation	ConvNet-S	Linear1, Linear2	20b
4.6	Task adaptation	ConvNet-D	Linear2, Linear3	20b

Table A.5

The dataset sizes for the transfer learning demonstration cases.

Section	Demonstration Case	Dataset Size	
		Offline Learning	Transfer Learning
4.3	Adaptation to the changes in data distribution	1024	1024
4.4	Adaptation to a domain extension	512	1024
4.5	Adaptation to a noisy data domain	1024	10240
4.6	Task adaptation	1024	1024

Table A.6

The dataset types for the transfer learning demonstration cases.

Section	Demonstration Case	Dataset Type	
		Offline Learning	Transfer Learning
4.3	Adaptation to the changes in data distribution	RANS + SA	RANS + RSM
4.4	Adaptation to a domain extension	RANS + SA	RANS + SA
4.5	Adaptation to a noisy data domain	RANS + SA	RANS + SA (noise added)
4.6	Task adaptation	RANS + SA	RANS + SA

References

- [1] W. Gracey, Summary of methods of measuring angle of attack on aircraft, 1958, National Advisory Committee for Aeronautics, Technical Note 4351.
- [2] F. Steinle, E. Stanewsky, Wind tunnel flow quality and data accuracy requirements, 1982, NATO AGARD Advisory Report No. 184, <https://apps.dtic.mil/sti/tr/pdf/ADA129881.pdf>.
- [3] R. Watzlavick, J. Crowder, F. Wright, Comparison of model attitude systems - active target photogrammetry, precision accelerometer, and laser interferometer, in: Advanced Measurement and Ground Testing Conference, 1996, pp. 1–12. <https://doi.org/10.2514/6.1996-2252>
- [4] B.M. Pedersen, T.F. Pedersen, H. Klug, N. van der Borg, N. Kelley, J.A. Dablberg, Expert group study on recommended practices for wind turbine testing and evaluation 11: Wind speed measurement and use of cup anemometry, 1999, Submitted to the Executive Committee of the International Energy Agency Programme for Research and Development on Wind Energy Conversion Systems. Edited by R.S.Hunter.
- [5] J.E. Zeis Jr, Angle of attack and sideslip estimation using an inertial reference platform, Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 1988. ADA194876, Available at <https://apps.dtic.mil/sti/tr/pdf/ADA194876.pdf>.
- [6] W. Gracey, Measurement of aircraft speed and altitude, 1958, NASA Reference Publication: 1046.
- [7] J.B. Barlow, W.H. Rae, A. Pope, Low-speed wind tunnel testing, John Wiley & Sons, New York, 3rd edition, New York, 1999.
- [8] J. Valasek, J. Harris, S. Pruchnicki, M. McCrink, J. Gregory, D.G. Sizoo, Derived angle of attack and sideslip angle characterization for general aviation, J. Guid. Contr. Dyn. 43 (6) (2020) 1039–1055. <https://doi.org/10.2514/1.G004010>
- [9] T. Finley, P. Tchong, Model attitude measurements at NASA Langley Research Center, in: Proc. of the 30th Aerospace Sciences Meeting and Exhibit, AIAA, 1992, pp. 1–7.
- [10] G. Lee, Study of optical techniques for the Ames unitary wind tunnels part 3. Angle of attack, 1992, NASA-CR-190541, Progress Report by MCAT Institute.
- [11] R.K. Bogue, H.W. Jentink, Optical air flow measurements in flight, 2004, NASA/TP-2004-210735.
- [12] B. Crawford, Angle measurement system (AMS) for establishing model pitch and roll zero, and performing single axis angle comparisons, in: Proc. of the 45th AIAA Aerospace Sciences Meeting and Exhibit, 2007, pp. 1–10. <https://doi.org/10.2514/6.2007-1162>
- [13] K.G. Toro, Technology review of wind-tunnel angle measurement, in: Proc. of the 11th International Symposium on Strain-Gauge Balances, 2018, pp. 1–10.
- [14] A. Lerro, A. Brandl, P. Gili, Model-free scheme for angle-of-attack and angle-of-sideslip estimation, J. Guid. Control Dyn. 44 (3) (2021) 595–600. <https://doi.org/10.2514/1.G005591>
- [15] M.S. Selig, B.D. McGranahan, Wind tunnel aerodynamic tests of six airfoils for use on small wind turbines: Period of performance: October 31, 2002-January 31, 2003, Technical Report NREL/SR-500-34515, National Renewable Energy Laboratory, 2004.
- [16] J. Bridges, M.P. Wernet, The NASA subsonic jet particle image velocimetry (PIV) dataset, 2011, Report/Patent Number E-17439, NASA/TM-2011-216807.
- [17] T. Narbuntas, S. Conley, K. Kallstrom, A.M. Abrego, Wind tunnel performance test of a ducted fan system, 2020, NASA/TM-2020-220484.
- [18] N.T. Smith, J.T. Heineck, E.T. Schairer, Optical flow for flight and wind tunnel background oriented Schlieren imaging, in: Proc. of the 55th AIAA Aerospace Sciences Meeting, 2017, pp. 1–18. <https://doi.org/10.2514/6.2017-0472>
- [19] T.J. Rohloff, S.A. Whitmore, I. Catton, Fault-tolerant neural network algorithm for flush air data sensing, J. Aircr. 36 (3) (1999) 541–549. <https://doi.org/10.2514/2.2489>
- [20] J. Quindlen, J. Langelaan, Flush air data sensing for soaring-capable UAVs, in: Proc. of the 51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2013, pp. 1–17. <https://doi.org/10.2514/6.2013-1153>
- [21] O.T. Topac, S.Y. Sara Ha, X. Chen, L. Gamble, D. Inman, F.-K. Chang, Hybrid models for situational awareness of an aerial vehicle from multimodal sensing, AIAA J. 61 (1) (2023) 305–314. <https://doi.org/10.2514/1.J061926>
- [22] P. Goupil, Airbus state of the art and practices on FDI and FTC in flight control system, Contr. Eng. Pract. 19 (2011) 524–539. <https://doi.org/10.1016/j.conengprac.2010.12.009>
- [23] D. Ossmann, H.D. Joos, P. Goupil, Enhanced sensor monitoring to maintain optimal aircraft handling in case of faults, J. Guid. Contr. Dyn. 40 (2017) 3127–3137. <https://doi.org/10.2514/1.G002341>
- [24] A. Vitale, F. Corrado, N. Genito, L. Garbarino, L. Verde, An innovative angle of attack virtual sensor for physical-analytical redundant measurement system applicable to commercial aircraft, Adv. Sci. Technol. Eng. Syst. J. 6 (1) (2021) 698–709. <https://doi.org/10.25046/aj060176>
- [25] B.W. Mersha, D.N. Jansen, H. Ma, Angle of attack prediction using recurrent neural networks in flight conditions with faulty sensors in the case of F-16 fighter jet, Complex Intell. Syst. 9 (2023) 2599–2611. <https://doi.org/10.1007/s40747-021-00612-6>
- [26] S.L. Brunton, J.N. Kutz, Promising directions of machine learning for partial differential equations, Nat. Comput. Sci. 4 (2024) 483–494. <https://doi.org/10.1038/s43588-024-00643-2>
- [27] D. Schwamborn, T. Gerhold, R. Heinrich, The DLR TAU-Code: Recent applications in research and industry, in: Proc. of the ECCOMAS CFD 2006 Conference, 2006, pp. 1–25. <https://elib.dlr.de/22421/>.
- [28] X.-Y. Gao, Two-layer-liquid and lattice considerations through a (3+1)-dimensional generalized Yu-Toda-Sasa-Fukuyama system, Appl. Math. Lett. 152 (2024) 109018. <https://doi.org/10.1016/j.aml.2024.109018>
- [29] H.D. Liu, B. Tian, Y.Q. Chen, C.D. Cheng, X.T. Gao, N-Soliton, Hth-order breather, hybrid and multi-pole solutions for a generalized variable-coefficient Gardner equation with an external force in a plasma or fluid, Nonlinear Dyn. 113 (2025) 3655–3672. <https://doi.org/10.1007/s11071-024-10397-1>
- [30] G. Wang, Z. Tan, X.-Y. Gao, J.-G. Liu, A new (2+1)-dimensional like-Harry-Dym equation with derivation and soliton solutions, Appl. Math. Lett. 172 (2026) 109720. <https://doi.org/10.1016/j.aml.2025.109720>
- [31] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (Volume 52, 2020) (2020) 477–508. <https://doi.org/10.1146/annurev-fluid-010719-060214>
- [32] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, D. Silver, Mastering Atari, Go, chess and shogi by planning with a learned model, Nature 588 (2020) 604–609. <https://doi.org/10.1038/s41586-020-03051-4>
- [33] T. Brown, B. Mann, N. Ryder, et al., Language models are few-shot learners, in: Advances in Neural Information Processing Systems, 33, Curran Associates, Inc., 2020, pp. 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc94967418bfb8ac142f64a-Paper.pdf.
- [34] J. Jumper, R. Evans, A. Pritzel, et al., Highly accurate protein structure prediction with AlphaFold, Nature 596 (2021) 583–589. <https://doi.org/10.1038/s41586-021-03819-2>
- [35] J.C. Ross, C.C. Jorgenson, M. Norgaard, Reducing wind tunnel data requirements using neural networks, 1997, Technical Memorandum, NASA TM-112193.
- [36] C.F. Lo, J.L. Zhao, R. DeLoach, Application of neural networks to wind tunnel data response surface methods, in: Proc. of the 21st AIAA Aerodynamic Measurement Technology and Ground Testing Conf., 2000, pp. 1–13.
- [37] T. Rajkumar, J. Bardina, Prediction of aerodynamic coefficients using neural networks for sparse data, in: Proc. of the AAAI FLAIRS Conf., 2002, pp. 242–246.
- [38] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: KDD'16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 1–10. <https://doi.org/10.1145/2939672.2939738>
- [39] E. Yilmaz, B. German, A convolutional neural network approach to training predictors for airfoil performance, in: Proc. of the 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2017, pp. 1–19. <https://doi.org/10.2514/6.2017-3660>
- [40] S. Lee, D. You, Prediction of laminar vortex shedding over a cylinder using deep learning, 2017, <https://arxiv.org/abs/1712.07854>. 1712.07854
- [41] J. Nagawkar, L. Leifsson, Multifidelity aerodynamic flow field prediction using random forest-based machine learning, Aerosp. Sci. Technol. 123 (2022) 107449. <https://doi.org/10.1016/j.ast.2022.107449>
- [42] M. Morimoto, K. Fukami, K. Zhang, K. Fukagata, Generalization techniques of neural networks for fluid flow estimation, Neural Comput. Appl. 34 (2022) 3647–3669. <https://doi.org/10.1007/s00521-021-06633-z>
- [43] D. Hines, P. Bekemeyer, Graph neural networks for the prediction of aircraft surface pressure distributions, Aerosp. Sci. Technol. 137 (2023) 108268. <https://doi.org/10.1016/j.ast.2023.108268>
- [44] S. Wassing, S. Langer, P. Bekemeyer, Physics-informed neural networks for parametric compressible Euler equations, Comput. Fluids 270 (2024) 106164. <https://doi.org/10.1016/j.compfluid.2023.106164>
- [45] X. Yan, J. Zhu, M. Kuang, X. Wang, Aerodynamic shape optimization using a novel optimizer based on machine learning techniques, Aerosp. Sci. Technol. 86 (2019) 826–835. <https://doi.org/10.1016/j.ast.2019.02.003>
- [46] W. Chen, K. Chiu, M.D. Fuge, Airfoil design parameterization and optimization using Bézier generative adversarial networks, AIAA J. 58 (11) (2020) 4723–4735. <https://doi.org/10.2514/1.J059317>
- [47] E. Yilmaz, B. German, Conditional generative adversarial network framework for airfoil inverse design, in: Proc. of the AIAA AVIATION 2020 FORUM, 2020, pp. 1–18. <https://doi.org/10.2514/6.2020-3185>
- [48] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166. <https://doi.org/10.1017/jfm.2016.615>
- [49] I. Staffell, R. Green, How does wind farm performance decline with age?, Renew. Energy 66 (2014) 775–786. <https://doi.org/10.1016/j.renene.2013.10.041>
- [50] H. Lu, X. Gao, J. Yu, Q. Zhao, X. Zhu, W. Ma, J. Cao, Y. Wang, Analysis and prediction of incoming wind speed for turbines in complex wind farm: Accounting for meteorological factors and spatiotemporal characteristics of wind farm, Appl. Energy 381 (2025) 125135. <https://doi.org/10.1016/j.apenergy.2024.125135>
- [51] D. Wang, M. Xu, Z. Guangming, F. Luo, J. Gao, Y. Chen, Enhancing wind power forecasting accuracy through LSTM with adaptive wind speed calibration (C-LSTM), Sci. Rep. 15 (2024) 5352. <https://doi.org/10.1038/s41598-025-89398-y>
- [52] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks, in: Proc. of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1717–1724. <https://doi.org/10.1109/CVPR.2014.222>
- [53] Y. Sun, S. Li, H. Gao, X. Zhang, J. Lv, W. Liu, Y. Wu, Transfer learning: A new aerodynamic force identification network based on adaptive EMD and soft thresholding in hypersonic wind tunnel, Chin. J. Aeronaut. 36 (8) (2023) 351–365. <https://doi.org/10.1016/j.cja.2023.03.024>
- [54] S. Krishnan, S. Ghosh, S. Atkinson, V. Andreoli, T. Vanputte, L. Wang, Transfer learning based modeling of industrial turbine airfoil characteristics, in: Proc. of the AIAA SCITECH 2022 Forum, 2022. <https://doi.org/10.2514/6.2022-2105>
- [55] Z. Wang, X. Liu, J. Yu, H. Wu, H. Lyu, A general deep transfer learning framework for predicting the flow field of airfoils with small data, Comput. Fluids 251 (2023) 105738. <https://doi.org/10.1016/j.compfluid.2022.105738>
- [56] R. Li, Y. Zhang, H. Chen, Transfer learning from two-dimensional supercritical airfoils to three-dimensional transonic swept wings, Chin. J. Aeronaut. 36 (9) (2023)

- 96–110. <https://doi.org/10.1016/j.cja.2023.04.008>
- [57] L. Cappugi, A. Castorrini, A. Bonfiglioli, E. Minisci, M.S. Campobasso, Machine learning-enabled prediction of wind turbine energy yield losses due to general blade leading edge erosion, *Energy Convers. Manage.* 245 (2021) 114567. <https://doi.org/10.1016/j.enconman.2021.114567>
- [58] A. Vaiuso, G. Immordino, M. Righi, A. Da Ronch, Multi-fidelity transonic aerodynamic loads estimation using Bayesian neural networks with transfer learning, *Aerosp. Sci. Technol.* 163 (2025) 110301. <https://doi.org/10.1016/j.ast.2025.110301>
- [59] C.M.A. Roelofs, C. Gück, S. Faulstich, Transfer learning applications for autoencoder-based anomaly detection in wind turbines, *Energy AI* 17 (2024) 100373. <https://doi.org/10.1016/j.egyai.2024.100373>
- [60] S.J. Pan, Q. Yang, A survey on transfer learning, *IEEE Trans. Knowl. Data Eng.* 22 (10) (2010) 1345–1359. <https://doi.org/10.1109/TKDE.2009.191>
- [61] L.Y. Pratt, J. Mostow, C.A. Kamm, Direct transfer of learned information among neural networks, in: *Proc. of the Ninth National Conference on Artificial Intelligence - Volume 2*, AAAI'91, AAAI Press, 1991, pp. 584–589.
- [62] D. Naik, R. Mammone, *Learning by learning in neural networks*, 24, Chapman and Hall, London, 1993.
- [63] T.M. Mitchell, S. Thrun, Learning one more thing, 1994, Technical Report, CMU-CS-94-184, CMU.
- [64] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: *Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, 2019, p. 4171–4186. <https://aclanthology.org/N19-1423.pdf>
- [65] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: *Proc. of the European Conference on Computer Vision – ECCV 2014*, Springer International Publishing, 2014, pp. 818–833.
- [66] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE* 86 (11) (1998) 2278–2324. <https://doi.org/10.1109/5.726791>
- [67] V. Mnih, K. Kavukcuoglu, D. Silver, et al., Human-level control through deep reinforcement learning, *Nature* 518 (2015) 529–533. <https://doi.org/10.1038/nature14236>
- [68] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: *Medical Image Computing and Computer-Assisted Intervention – MICCAI'2015*, Springer International Publishing, Cham, 2015, pp. 234–241.
- [69] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, real-time object detection, in: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
- [70] I. Goodfellow, Y. Bengio, A. Courville, *Deep learning*, MIT Press, 2016. <http://www.deeplearningbook.org>
- [71] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: *Proc. of the Advances in Neural Information Processing Systems*, 25, Curran Associates, Inc., 2012, pp. 1–9. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
- [72] T. maintainers, contributors, TorchVision: PyTorch's computer vision library, 2016, (<https://github.com/pytorch/vision>).
- [73] J. Ansel, E. Yang, H. He, N. Gimselshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C.K. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M.Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, S. Chintala, PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation, in: *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Volume 2 (ASPLOS '24), ACM, 2024, pp. 1–18. <https://doi.org/10.1145/3620665.3640366>
- [74] R.J. Zwaan, Data Set 4: NLR 7301 supercritical airfoil oscillatory pitching and oscillating flap, 1982, NATO AGARD Report 702: Compendium of unsteady aerodynamic measurements, <https://www.sto.nato.int/publications/AGARD/AGARD-R-702/AGARD-R-702.pdf>
- [75] S.R. Allmaras, F.T. Johnson, P.R. Spalart, Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model, in: *Proc. of the 7th International Conference on Computational Fluid Dynamics, ICCFD7-1902*, Springer International Publishing, Big Island, Hawaii, 2012, pp. 1–11. https://www.iccfd.org/iccfd7/assets/pdf/papers/ICCFD7-1902_paper.pdf
- [76] B. Eisefeld, C. Rumsey, V. Togiti, Verification and validation of a second-moment-closure model, *AIAA J.* 54 (5) (2016) 1524–1541. <https://doi.org/10.2514/1.J054718>
- [77] P. Bekemeyer, A. Bertram, D.A.H. Chaves, M.D. Ribeiro, A. Garbo, A. Kiener, C. Sabater, M. Stradtner, S. Wassing, M. Widhalm, S. Goertz, F. Jaekel, R. Hoppe, N. Hoffmann, Data-driven aerodynamic modeling using the DLR SMARTy toolbox, in: *Proc. of the AIAA AVIATION 2022 Forum*, 2022, pp. 1–20. <https://doi.org/10.2514/6.2022-3899>
- [78] J.H. Halton, Algorithm 247: Radical-inverse quasi-random point sequence, *Commun. ACM* 7 (12) (1964) 701–702. <https://doi.org/10.1145/355588.365104>