applied sciences

MDPI

Article

Fast Prediction of Flow Field around Airfoils Based on Deep Convolutional Neural Network

Ming-Yu Wu [1], Yan Wu [2], Xin-Yi Yuan [2], Zhi-Hua Chen [1], Wei-Tao Wu [2,*] and Nadine Aubry [3,*]

[1] Key Laboratory of Transient Physics, Nanjing University of Science and Technology, Nanjing 210094, China
[2] School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[3] Department of Mechanical Engineering, Tufts University, Medford, MA 02155, USA
* Correspondence: weitaowwtw@njust.edu.cn (W.-T.W.); nadine.aubry@tufts.edu (N.A.)

Abstract: We propose a steady-state aerodynamic data-driven method to predict the incompressible flow around airfoils of NACA (National Advisory Committee for Aeronautics) 0012-series. Using the Signed Distance Function (SDF) to parameterize the geometric and flow condition setups, the prediction core of the method is constructed essentially by a consecutive framework of a convolutional neural network (CNN) and a deconvolutional neural network (DCNN). Impact of training parameters on the behavior of the proposed CNN-DCNN model is studied, so that appropriate learning rate, mini-batch size, and random deactivation rate are specified. Tested by "unseen" airfoil geometries and far-field velocities, it is found that the prediction process is three orders of magnitudes faster than a corresponding Computational Fluid Dynamics (CFD) simulation, while relative errors are maintained lower than 1% on most of the sample points. The proposed model manages to capture the essential dynamics of the flow field, as its predictions correspond reasonably with the reconstructed field by proper orthogonal decomposition (POD). The performance and accuracy of the proposed model indicate that the deep learning-based approach has great potential as a robust predictive tool for aerodynamic design and optimization.

Keywords: convolutional neural network; deconvolutional neural network; signed distance function; flow field prediction; deep learning

Citation: Wu, M.-Y.; Wu, Y.; Yuan, X.-Y.; Chen, Z.-H.; Wu, W.-T.; Aubry, N. Fast Prediction of Flow Field around Airfoils Based on Deep Convolutional Neural Network. *Appl. Sci.* **2022**, *12*, 12075. https://doi.org/10.3390/app122312075

Academic Editor: Wei Huang

Received: 21 October 2022
Accepted: 23 November 2022
Published: 25 November 2022

1. Introduction

The aerodynamic performance and safety of airfoil rely on multiple parameters including geometry profile, flight speed, angle of attack (AOA), atmospheric density, and other operating conditions. As parameter study by wind tunnel experiment is prohibitively expensive and time-consuming, a more efficient approach to evaluate the aerodynamic performance is generally required, especially during the design process, so that choices of parameters could be narrowed before physical experiments. Currently, the most typical solution for fast aerodynamic evaluation of is Computational Fluid Dynamics (CFD), when solving Reynolds-averaged Navier–Stokes (RANS) equations, the computational cost of CFD is significantly lower than that of a wind tunnel test at the same scale. However, as optimization metrics of modern aircraft design become more sophisticated, the volume of parameter studies also grows substantially, in which case even RANS becomes too expensive, and an evaluation or prediction method cheaper than CFD is necessary [1–3]. One important solution for fast aerodynamic evaluation, such as flow fields, pressure distribution coefficient, lift coefficient, and drag coefficient, is to constitute surrogate models. The basic idea is to run CFD calculations on only a few sampled geometry profiles, train and construct a surrogate model through the CFD database, then use the surrogate model instead of explicit CFD for further calculations of airfoil geometries outside the sampled database. As surrogate models avoid the expensive solution of RANS equations, the prediction (calculation) wall time could be greatly reduced, making the search for optimal geometry much more practical under variable flow conditions. Major developments of the

surrogate model coupled with CFD include methods of multi-fidelity, reduced-order model (ROM), and data-driven model [4]. They have been successfully applied to enhance the efficiency of aerodynamic performance evaluation, such as polynomial response surface (PRS) [5], artificial neural network (ANN) [6,7], radial basis function (RBF) [8], support vector machine (SVM) [9], Kriging model [10–12], and Gaussian process (GP) [13,14] regression. Nevertheless, most of the traditional surrogate models are narrowly appropriate for predicting low-dimensional physical quantities such as lift and drag coefficient, and moment coefficient, offering averaged characteristics rather than the complete high-dimensional flow field information.

Fast development of machine learning (ML) technology in recent years bring promising breakthroughs for surrogate models to directly address high-dimensional physical fields. Santos et al. [15] and Wallach et al. [16] applied the neural network method to generic aircraft to predict aerodynamic coefficients which are modeled as functions of the angle of attack (AOA), Mach number, Reynolds number, and airfoil geometry. Wu et al. [17] conducted a benchmark study on the optimization of aerodynamic shape using POD-based Class-Shape Function Transformation (CST) methods. Raul and Leifsson developed a Kriging regression surrogate model-based optimization technique to mitigate the deep dynamic stall characteristics of airfoils [18]. Zhu et al. [19] replaced the solution of governing partial differential equations with neural networks, which directly reconstruct the mapping between the turbulent eddy viscosity and the mean flow variables. ML or neural network methods aim to establish a non-linear mapping between input parameters and output, saving the otherwise expensive solution of Navier Stokes equations. However, they just mechanically search for pure input-output mapping [20,21] without exploiting the abundant and intrinsic physical information which exactly concerned designers. Besides, most current ML-based surrogate models are shallow models, they are trained towards only the original function subspace, rather than the whole complex nonlinear function space, lacking generalization ability for data never seen from the training set. As a result, large amounts of data are required to be fed to the model, making it difficult to deal with high-dimensional nonlinear problems.

As an important branch of ML, deep learning methods are designed in the first place for high-dimensional nonlinear problems, typically coupled with massive data. Deep learning achieves great success in the area of computer vision, then rapidly attracts research attention in the field of structural and fluid mechanics [22,23]. Specifically for aerodynamic design and optimization, deep learning has been applied to flow prediction [24–29], flow field reconstruction [30,31], modes classification [32], fault detection [33], and design optimization [34]. Ling and Templeton, for example, utilized deep learning to identify regions of high uncertainty and drove the development of RANS turbulence modeling [35], it turns out that deep learning can effectively mine the deep information from the flow field data (obtained from expensive CFD calculations) and finish the predicting of the entire flow field in milliseconds.

In particular, the convolutional neural network (CNN) is widely used in the field of aerodynamic design and optimization to reduce the complexity of the deep learning network, through strategies of the local receptive field, weight sharing, and down-sampling, as it is invariant to the degree of variation in the form of translation, rotation, and scaling. Zhang et al. [36] trained two types of CNN architectures to predict the lift coefficient of a given airfoil shape in different flow conditions, the results demonstrate that their CNN model has good accuracy and could learn faster than the multi-layer perceptron (MLP). Yu et al. presented another CNN-based model to learn the airfoil lift coefficient calculated from CFD results, where a "feature-enhanced-image" data preprocessing method was developed to achieve higher prediction accuracy [37]. Hui et al. [38] proposed a CNN-based data-driven approach to quickly predict the pressure distribution around the airfoils with a mean square error of less than 2%. Peng et al. [39] used CNN to establish an unsteady data-driven ROM for predicting the velocity field around an airfoil, where a mapping function was constructed between the temporal evolution of the pressure and velocity. Li et al. [40]

adopted long-term memory (LSTM) network to build an unsteady aerodynamic ROM, as it could accurately capture the dynamic characteristics of aerodynamic systems under various flow and structure parameters. Wu et al. [41] built a CNN model combined with the generative adversarial network (GAN) to achieve the mapping from parameterized airfoils to corresponding transonic flow field under fixed operating conditions. Tompson et al. [42] developed a data-driven model to obtain real-time predictions of 2D and 3D flows with good efficiency and generalization property.

This work aims to propose a novel approach to achieve accurate and rapid flow field inference for airfoil design optimization based on deep learning, in a less computationally demanding manner instead of expensive CFD simulation tools. In this paper, a CNN-based data-driven method is proposed for fast prediction of the steady flow field around NACA (National Advisory Committee for Aeronautics) 0012 airfoil under different far-field velocities for a series of geometric parameters. The morphing of the airfoil profile is controlled by changing the maximum camber and its position along the chord, while keeping maximum thickness at a constant value. The model establishes a mapping function between the geometric profile and corresponding external flow fields, namely pressure, and velocity. Specifically, the proposed model consists of a CNN and a deconvolutional neural network (DCNN), referred to as the CNN-DCNN model. Acting as an encoder, CNN mines the intrinsic features of the physical field [43] into a lower dimensional space, while DCNN decodes and reconstructs the full-dimensioned physical field. To improve the performance of the model, the Signed Distance Function (SDF) is used to parameterize the geometric parameters of the airfoil profile. In a word, the main contributions of this work are as follows:

(1) We designed a data-driven reduced-order model called CNN-DCNN model based on deep learning, which can predict the pressure and velocity field around airfoils with a mean relative error lower than 1%, costing only 25 ms, three orders of magnitudes faster than CFD.
(2) SDF that is a universal and flexible parameterization method was adopted to represent the shape information of airfoil geometries. Further, to make the model adaptive to different flow conditions, we stack the operating parameters, i.e., Reynolds number with the SDF matrix to form the input of the CNN-DCNN model.
(3) We compare the CNN-DCNN model with the pure numerical method, namely, POD, in extracting the essential features of the flow field, enhancing the interpretability of the data-driven reduced-order model based on deep learning.

The structure of this paper is organized as follows. In Section 2, the overall architecture of the CNN-DCNN framework is given, followed by more detailed introductions on the structure and sub-layers of the prediction function. Training and evaluation methods are also presented, along with numerical methods and data preparation, including the CFD validation, the airfoil geometry representation by SDF, and data preparation and processing. In Section 3, we first explore the effect of train parameters on the performance of the CNN-DCNN model, then the accuracy of its predicted results is measured and discussed. We further apply POD (proper orthogonal decomposition) analysis on pressure and velocity fields from both CFD and predicted outputs, to quantify the feature extraction capability of the proposed model, whose extensionality to unseen flow fields is then investigated with Reynolds numbers (far-field velocities) falls outside tested range. Section 4 summarizes the current work.

2. Methods

This paper focuses on constructing a data-driven prediction model based on supervised deep learning, mapping the SDF of various airfoil geometries to the corresponding pressure and velocity fields. In general, the proposed model applies a CNN to encode and extract features of airfoil geometry and flow conditions, then a DCNN is utilized to decode the features and map them back to the pressure and velocity field.

2.1. Design of Neural Network Model

2.1.1. Overall Architecture of CNN-DCNN Framework

Figure 1 presents the overall architecture of the CNN-DCNN framework, which aims to establish a deep learning-based neural network to regress a nonlinear function as follows:

$$(\hat{P},\ \hat{U}) = f(S, Re; W, b) \tag{1}$$

$$Re = cV_{ref}/\mu \tag{2}$$

where P and U are the pressure and velocity field (specifically, U represents the velocity magnitude), while the $\hat{}$ hat are placed on fields predicted by CNN-DCNN model, S is the input SDF matrix containing the information of AOA and airfoil shape, W and b are the network parameters to learn, whose updating rules will be introduced in Section 2.2. W is the weight parameter matrix, and b is the bias parameter matrix. Re is the Reynolds number defined by far field incoming velocity V_{ref}, chord length c of airfoil, and dynamic viscosity μ. The main prediction process of CNN-DCNN is accomplished by the nonlinear function f, which is expected to approximate, under different design parameters expressed by SDF matrix, the steady-state $\hat{P}$ and $\hat{U}$ as closely as possible to the P and U given by CFD simulations. In fact, the functional expression f The working procedure of the CNN-DCNN framework shall be summarized into three steps, namely data generation, training, and testing:

(I) Data generation: Firstly, a geometry library of airfoil is built up, including AOA, maximum camber, and its position along the chord. Those parameters are then formulated into input S by SDF, while corresponding CFD meshes are generated through gmsh, so that the flow field (P, U) could be obtained by CFD calculation under given operating parameters, i.e., Reynolds number. As the result, the input $(S,\ Re)$ and the label $(P,\ U)$ compose the dataset $(S,\ Re,\ P,\ U)$.

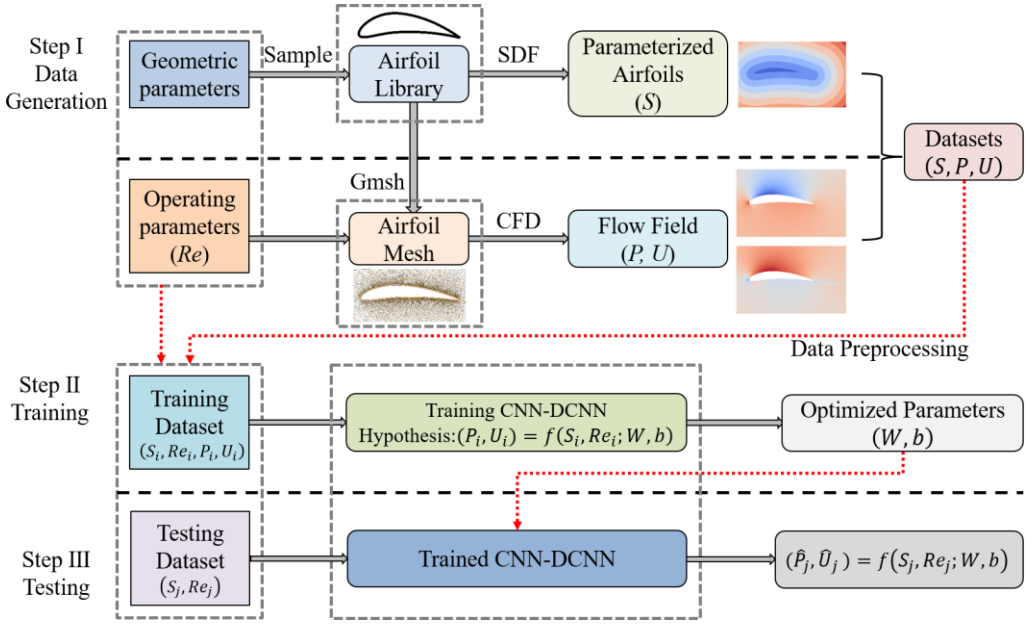

Figure 1. Overall architecture of CNN-DCNN framework.

(II) Training: Saving a small portion as further "testing dataset", the dataset $(S,\ Re,\ P,\ U)$ acquired in step I is then split into "training dataset" and "validation dataset". (Validation dataset is used to judge whether the model is overfitting [44]) Through the combination of training dataset and operating parameters, the CNN-DCNN model is trained towards the hypothesis of $(P, U) = f(S,\ Re; W, b)$ as in Equation (1).

(III) Testing: The trained model is tested by predicting $(\hat{P},\ \hat{U})$ of input S from the unseen airfoils, provided by the testing dataset. Accuracy of the CNN-DCNN model is then measured by comparing $(\hat{P},\ \hat{U})$ with (P, U).

2.1.2. Design of the CNN-DCNN Prediction Model

The CNN-DCNN model for predicting the flow field mentioned above is the core of the CNN-DCNN framework, whose specific architecture is displayed in Figure 2. Mathematically speaking, the model works as a nonlinear function that maps the low-dimensional features extracted by the CNN encoder to the high dimensional output (e.g., the 2D flow field in this work). Therefore, we first need to extract abundant features among the input variables (flow conditions and airfoil geometries) and assign them appropriate weights, then the input variables can be well mapped to the output flow field. As matter of fact, the 2D flow field data is treated as 2D images in the mapping process, so the mathematical tools developed for CNN and DCNN in computer vision industry could be utilized. The CNN-DCNN model is composed of a series of six encoder (CNN) layers, six decoder (DCNN) layers and a one-by-one convolutional layer. The last layer is introduced as a nonlinear activation of the previous layer, to enhance the network nonlinear expression. In this sense, the prediction model employs CNN to extract information on airfoil shape and operating parameters from the input data, encode them to feature properties, and then utilizes DCNN to decode these features and predict the flow field. In general, the learning process of the CNN-DCNN based prediction model could be abstracted as:

$$F = encode(S,\ Re),\ (\hat{P},\ \hat{U}) = decode(F) \tag{3}$$

where S is the SDF representation of the parameterized airfoil geometry, F is the features mined from the input $(S,\ Re)$ using CNN encoder, such as the information of airfoil profile and the flow conditions (i.e., Reynolds number). Furthermore, since the airfoil profile is the actual coordinates after rotational transformation, F also includes the information on the angle of attack (AOA). $\hat{P}$, $\hat{U}$ represent the pressure and velocity field predicted from the features F by DCNN decoder. In essence, the CNN encoder compresses the high-dimensional input states to the feature maps, finding out the reduced dimensional representation of the input matrices, while DCNN optimally removes the channel-wise and pixel-wise correlations before these feature maps fed into each layer, and projects these feature maps to output matrices. Details of the training process for filter parameters of CNN and DCNN are given in Section 2.2.

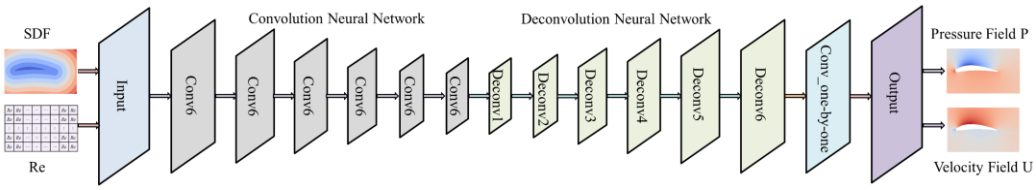

Figure 2. Specific architecture of flow field prediction network. "Conv" denotes convolutional layer, "Deconv" indicates deconvolutional layer, "Conv_one-by-one" refers to the one-by-one convolutional layer.

Model parameters of all the sub-layers are given in Table 1, where $n_w \times n_h \times n_c$ is the size of the filter (w: width, h: height, c: channel), n_F is the number of the filters ($F : filter$), where following the descriptive convention of convolutional networks, we refer to a 3D cube stacked with multiple 2D convolutional kernels as a filter in this work. s is the stride of the convolution, and $N_w \times N_h \times d$ is the size of the output matrix of current layer. Specifically, the number of filters n_F is decided by the index of the CNN layer by 2^{3+l}. To obtain sufficient depth and enhance the nonlinear mapping capability of the network,

activation function of "Exponential Linear Unit (ELU)" is applied on each hidden layer. The form of ELU activation function is given as

$$g(x) = \begin{cases} x, & x \geq 0 \\ \mu(e^x - 1), & x < 0 \end{cases} \tag{4}$$

where μ is the default coefficient typically set to unity. Behavior of Equation (4) makes sure that the output of ELU activation function is one-sided saturated and obeys a distribution with zero mean, which speeds up the convergence of training process. Introduction of ELU also avoids the gradient vanishing and exploding in deep neural networks by applying identity function for all nonnegative arguments. As the negative input can also be activated by ELU, which further prevents the deactivation of the neurons during the network training.

Table 1. Model Parameters of convolutional layers.

Layer Name	Executing Operation	Shape
	$n_w \times n_h \times n_c \times n_F/s$	$N_w \times N_h \times d$
Input	—	$208 \times 340 \times 2$
Conv1	$4 \times 8 \times 2 \times 16/2$	$103 \times 167 \times 16$
Conv2	$5 \times 7 \times 16 \times 32/2$	$50 \times 81 \times 32$
Conv3	$4 \times 7 \times 32 \times 64/2$	$24 \times 38 \times 64$
Conv4	$4 \times 8 \times 64 \times 128/2$	$11 \times 16 \times 128$
Conv5	$3 \times 4 \times 128 \times 256/2$	$5 \times 7 \times 256$
Conv6	$2 \times 3 \times 256 \times 512/1$	$4 \times 5 \times 512$
Deconv1	$2 \times 3 \times 512 \times 256/1$	$5 \times 7 \times 256$
Deconv2	$3 \times 4 \times 256 \times 128/2$	$11 \times 16 \times 128$
Deconv3	$4 \times 8 \times 128 \times 64/2$	$24 \times 38 \times 64$
Deconv4	$4 \times 7 \times 64 \times 32/2$	$50 \times 81 \times 32$
Deconv5	$5 \times 7 \times 32 \times 16/2$	$103 \times 167 \times 16$
Deconv6	$4 \times 8 \times 16 \times 2/2$	$208 \times 340 \times 2$
Conv_one-by-one	$1 \times 1 \times 2 \times 2/1$	$208 \times 340 \times 2$
Output	Reshape	$208 \times 340 \times 2$

2.1.3. Convolutional Layer

The convolutional layer of CNN executes most of the computational promotion. Figure 3 is a typical convolution scheme where the local receptive field with a kernel of 3×4 maps to 3 filters. During the convolution process, the filter slides across the whole input with a certain stride, meanwhile, the element-wise product is executed, with homologous output responses at each spatial position. The size of the output after the convolutional operation is calculated as:

$$N_w^{[l+1]} = \frac{N_w^{[l]} - n_w^{[l+1]} + 2p_w^{[l+1]}}{s^{[l+1]}} + 1 \tag{5}$$

$$N_h^{[l+1]} = \frac{N_h^{[l]} - n_h^{[l+1]} + 2p_h^{[l+1]}}{s^{[l+1]}} + 1 \tag{6}$$

where the superscript l and $l+1$ represent the index of the convolutional layers (could be called by lth and $(l+1)$th convolutional layer later), the subscript w and h refer to the width and height of the kernel, while p is the padding size. Taking the change of N_w from "Conv2" to "Conv3" for example, there is $24 = (50 - 4 + 0)/2 + 1$ (no padding). We can also see from Figure 3 that the size of the convolution kernel ($n_w \times n_h \times n_c$) is $3 \times 4 \times 1$, the number of filters n_F is 3, after the convolution operation, the input image of 11×22 is compressed into the output of shape $9 \times 19 \times 3$ according to Equations (5) and (6). In addition, the index of the slide window (box in red solid line) is row 6 and column 9, after the convolution process (point-to-point multiplication with the convolution kernel

followed by summation), it is projected to the corresponding position of 3 filters. Moreover, the local operation of a given convolutional layer is carried out as follows:

$$y = \sum_i \sum_j w_i x_j + b \tag{7}$$

where y is the local output, w_i is the weight coefficient of kernel i, x_j is the local receptive field seen by the neurons, and b is the bias matrix. Subsequently, the ELU activation function (see Equation (4)) will be applied to obtain the output of the current layer. Actually, the weight parameter refers to the actual values associated with each element and indicates the importance of that element in predicting the final value, which can be understood as the coefficients of each term of a nonlinear function. As described in Equation (7), the bias parameter is a threshold value to shift the activation function to the left or right, which can be called the intercept distance along the y direction in the linear equation. Finally, the f in Equation (1) can be summarized as follows provided that l indicates the last layer:

$$f = W^l \times g\left(y^{l-1}\right) + b^l \tag{8}$$

where y^{l-1} is the output $(l-1)$th layer, $g\left(y^{l-1}\right)$ denotes the final output after using activation function, and W^l and b^l represent the weight and bias matrix of the lth layer, respectively.

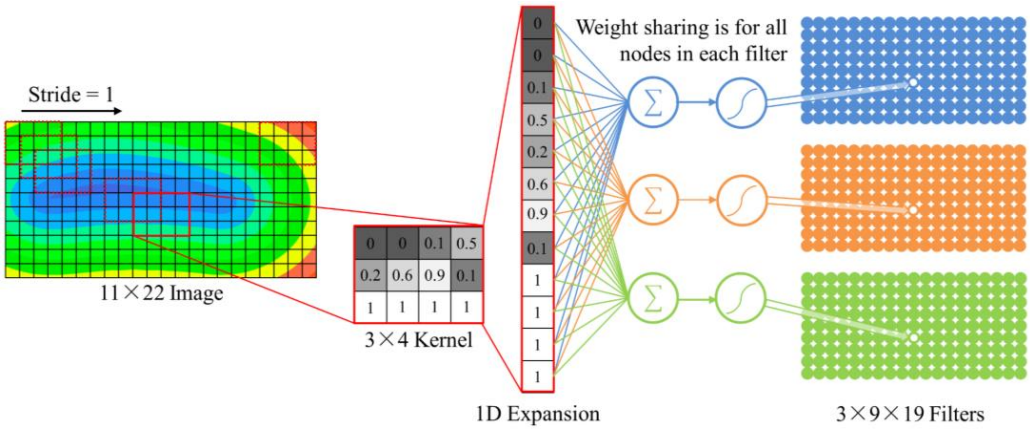

Figure 3. The schematic of a convolutional layer (numbers are for illustration purposes only; bule, orange, and green represent different filters).

To reduce the number of parameters and speed up the convolution process, local connectivity and weight sharing schemes are employed in each convolutional layer. The local connectivity is utilized to connect the local receptive field of the input to each neuron to learn the local features. While weight sharing scheme is to keep using identical convolving parameters for all convolutional kernels within a filter, to detect the same features from different regions, but apply different weight parameters for each kernel to detect different features. Besides local connectivity and weight sharing schemes, the activation function is applied after the convolutional layer to make the network deep enough and enhance the nonlinear fitting ability.

2.1.4. Deconvolutional Layer

While the convolutional layers are able to sample features from the input to reduce the dimensions of input information, it is often necessary to restore the information back to its original dimensions to complete the prediction function. This "up-sampling" procedure is typically realized by the deconvolution layers. Figure 4a demonstrates the computational procedure of a 2D deconvolution, where the input image of 2×2 with a kernel of 3×3 is mapped to the output of 5×5 through the element-wise multiplication between the input and the kernel. Figure 4b gives another example of a single deconvolutional network layer

applying to a 2D matrix: Random values padding is first performed through a special form of the convolution to increase the size of the input, then the same convolution operation is executed to adjust the width and the height of the output. The size of the output after deconvolution is inversely computed as follows:

$$N_w^{[l+1]} = \left(N_w^{[l]} - 1\right) \times s^{[l+1]} - 2p_w^{[l+1]} + n_w^{[l+1]} \tag{9}$$

$$N_h^{[l+1]} = \left(N_h^{[l]} - 1\right) \times s^{[l+1]} - 2p_h^{[l+1]} + n_h^{[l+1]} \tag{10}$$

In this sense, the output information could be synthesized by the deconvolution procedure, once the latent feature map is given. Then combined with suitable loss functions and train methods, high quality latent representations can be obtained by the deconvolutional layer.

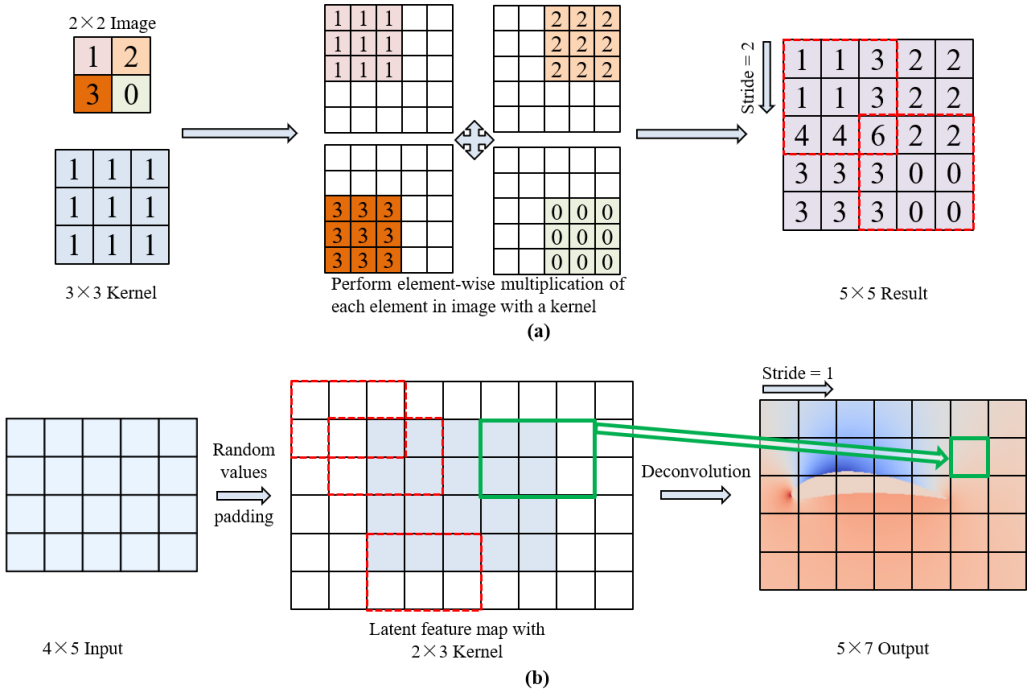

Figure 4. The schematic of a deconvolutional layer: (**a**) computation procedure of a 2D deconvolution; (**b**) operating flow of a deconvolution layer (numbers are for illustration purpose only).

2.2. Train and Evaluation Methods

In this work, the CNN-DCNN model is trained by supervised learning, in which the labeled data along with the input matrices are fed to network. In the training process, the model parameters are optimized iteratively towards the direction that minimizes the loss between the predicted results $(\hat{P}, \hat{U})$ and the labeled outputs (P, U). To obtain optimal parameters, loss is quantificationally by an L2 loss function with L2 regularization to avoid overfitting, the loss function is defined as

$$J = \frac{1}{2N} \sum_{n=1}^{N} \left[\left(P_n - \hat{P}_n\right)^2 + \left(U_n - \hat{U}_n\right)^2 \right] + \lambda \|W\|_2^2 \tag{11}$$

where P_n and U_n denote nth flow field results by CFD simulation, $\hat{P}_n$ and $\hat{U}_n$ represent the corresponding prediction by CNN-DCNN, N is the number of training data, λ is the

regularization coefficient, and $\lambda\|W\|_2^2$ performs the L2 regularization. To eliminate the effect from the airfoil interior points, the original loss function is modified as follows:

$$J' = \frac{1}{2N}\sum_{n=1}^{N}\left[\left(\left(P_n - \hat{P}_n\right)\times\delta_n\right)^2 + \left(\left(U_n - \hat{U}_n\right)\times\delta_n\right)^2\right] + \lambda\|W\|_2^2 \tag{12}$$

where δ_n is the binary mask matrix of the nth airfoil, which is filled with 0 and 1, for points inside and outside of the airfoil, respectively.

To minimize the loss function defined in Equation (12), the Adaptive Moment Estimation (Adam) algorithm is used to update the model parameters, due to its performance advantage [45]. For ease of description, W and b in Equation (1) are collectively referred to as network parameters and represented by θ since their updating rules are the same. The Adam algorithm employs both the first and second moments to improve the update of parameters:

$$V_{d\theta} = \beta_1 V_{d\theta} + (1-\beta_1)d\theta \tag{13}$$

$$S_{d\theta} = \beta_2 S_{d\theta} + (1-\beta_2)d\theta^2 \tag{14}$$

where θ represents the network's parameters set including the weight W and the bias b, $d\theta$ refers to the gradients of the loss function with respect to θ, β_1 and β_2 are the exponential decay rates of the moment estimation, $V_{d\theta}$ and $S_{d\theta}$ are the first and second moment estimations. Considering that both $V_{d\theta}$ and $S_{d\theta}$ are initialized by zero vectors and also bias towards zero during update, the following correction is carried out to adjust these biases within each training iteration,

$$V_{d\theta}^{corrected} = \frac{V_{d\theta}}{1-\beta_1^t} \tag{15}$$

$$S_{d\theta}^{corrected} = \frac{S_{d\theta}}{1-\beta_2^t} \tag{16}$$

where the superscript t of β is the exponential operator. Finally, the network parameters θ are updated in an iterative manner through

$$\theta = \theta - \alpha\frac{V_{d\theta}^{corrected}}{\sqrt{S_{d\theta}^{corrected}+\varepsilon}} \tag{17}$$

where α is the learning rate and ε is a small scalar to avoid singularity. Moreover, it is worth mentioning that the learning strategy of mini-batch gradient descent is applied in the training process to improve computational efficiency and model robustness [46]. Random deactivation rate (dropout) of 0.2 is further used for all layers to prevent overfitting.

To measure the prediction accuracy of the trained CNN-DCNN model, the relative error of the nth pressure field (PRE) and velocity field (URE) are defined as follows, respectively:

$$PRE = \frac{\|\hat{P}_n - P_n\|_F^2}{\|P_n\|_F^2}\times 100\% \tag{18}$$

$$URE = \frac{\|\hat{U}_n - U_n\|_F^2}{\|U_n\|_F^2}\times 100\% \tag{19}$$

Correspondingly, the prediction accuracy can be computed by subtracting PRE or URE from 1.

2.3. Numerical Method and Data Preparation

2.3.1. CFD Validation

To prepare the CFD datasets in this work, we use the open source software OpenFOAM for the solution of RANS equations. Before the buildup of the dataset, the CFD numerical simulation method to obtain the accurate flow field data will be validated by the calculation of steady flow around NACA0012 airfoil. The 2D unstructured mesh with the boundary layers near the airfoil is generated via gmsh from airfoil profile, where the used gmsh is a python-based application program interface (API). Considering the influence of the far-field boundary on the flow field around the airfoil and computational cost, the entire simulation domain is set to 40 c × 40 c, where c is the chord length of airfoil. Figure 5 depicts the schematic diagram of the calculation domain and mesh. The number of grids is approximately 5×10^4.

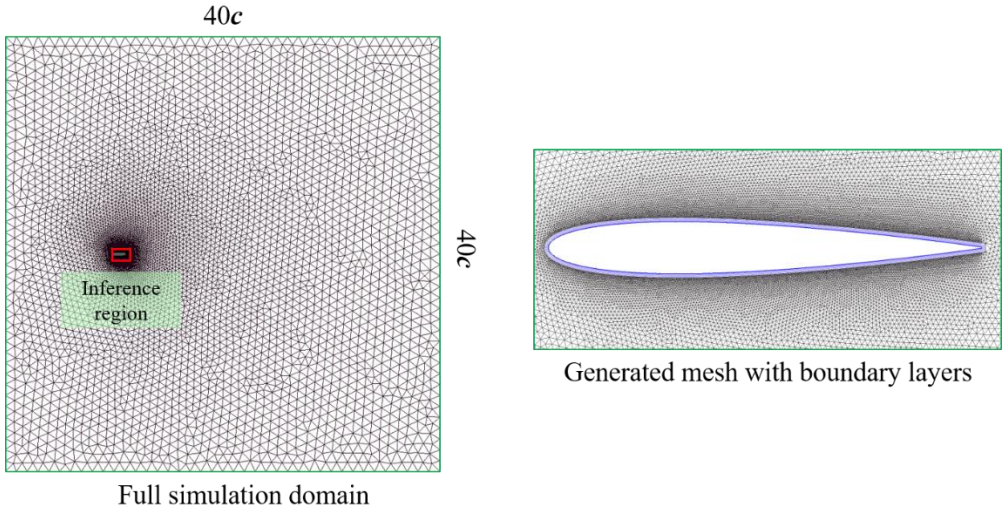

Figure 5. Schematic diagram of the calculation domain and mesh used in the validation case.

The Spalart–Allmaras (SA) turbulence model is selected to solve the RANS equations using OpenFOAM. The operating conditions of the validation case are as follows: the Mach number (Mach) is 0.15, Reynolds number is 6×10^6, AOA is 10°. Simulation results of the validation case are also compared with data from the numerical results of CFL3D (Computational Fluids Laboratory-3D) User's Manual [47] and wind tunnel experiment [48] in terms of lift coefficient C_l, drag coefficient C_d and pressure coefficient C_p, which are defined as

$$C_l = \frac{2f_l}{\rho U_\infty^2 c}, \; C_d = \frac{2f_d}{\rho U_\infty^2 c} \tag{20}$$

$$C_p = \frac{p - p_\infty}{1/2\rho U_\infty^2} \tag{21}$$

where f_l is the lift force perpendicular to the direction of incoming flow, and f_d is the drag force of the same direction of the incoming flow, p is the static pressure at the point where pressure coefficient is being evaluated; p_∞ is the static pressure in the freestream remote from any disturbance; ρ is the fluid density; and U_∞ is the inlet velocity.

Table 2 shows C_l and C_d from the simulation results of OpenFOAM, in comparison with CFL3D and wind tunnel experiment data. The relative error is calculated by taking the experimental data as the reference value. Admittedly, relative errors by simulation on the square grid are larger than CFL3D, especially in terms of drag coefficient C_d. Distribution of surface pressure coefficient C_p is given in Figure 6a, along both sides of the airfoil. The simulation results from OpenFOAM agree reasonably with the experiment results, with largest discrepancy takes place at the upper of leading edge. Moreover, the variations of C_l and C_d are shown in Figure 6b. When the grid number exceeds 5×10^4, the sensitivity of

lift and drag coefficients to the grid number is rather low. Moreover, with the grid number of 5×10^4 under the validation condition, the calculated C_l and C_d are 1.1283 and 0.0129, respectively. Slight deviations are achieved compared with the experimental values of 1.0809 and 0.0117. Therefore, the grid-scale of about 5×10^4 is selected for subsequent calculations. As the focus of this work is on the prediction and training behavior of the proposed CNN-DCNN model, rather than accuracy of the CFD method, we assume that results from the square grid with a size of 40 c × 40 c are acceptable for further training and testing of the CNN-DCNN model. Additionally, the flow still has laminar to turbulent transition at Reynolds number of millions of orders of magnitude [49]. Nevertheless, it should be emphasized that the final collected flow field data is under steady flow status. At this time, the flow has transformed to a fully turbulent status.

Table 2. Comparison of lift and drag coefficient from different data sources.

Data Sources	C_l	C_d	Relative Error	
			C_l	C_d
Experiment	1.0809	0.0117	-	-
CFL3D	1.0909	0.0123	0.93%	5.13%
OpenFOAM-40 × 40	1.1283	0.0129	4.39%	10.3%

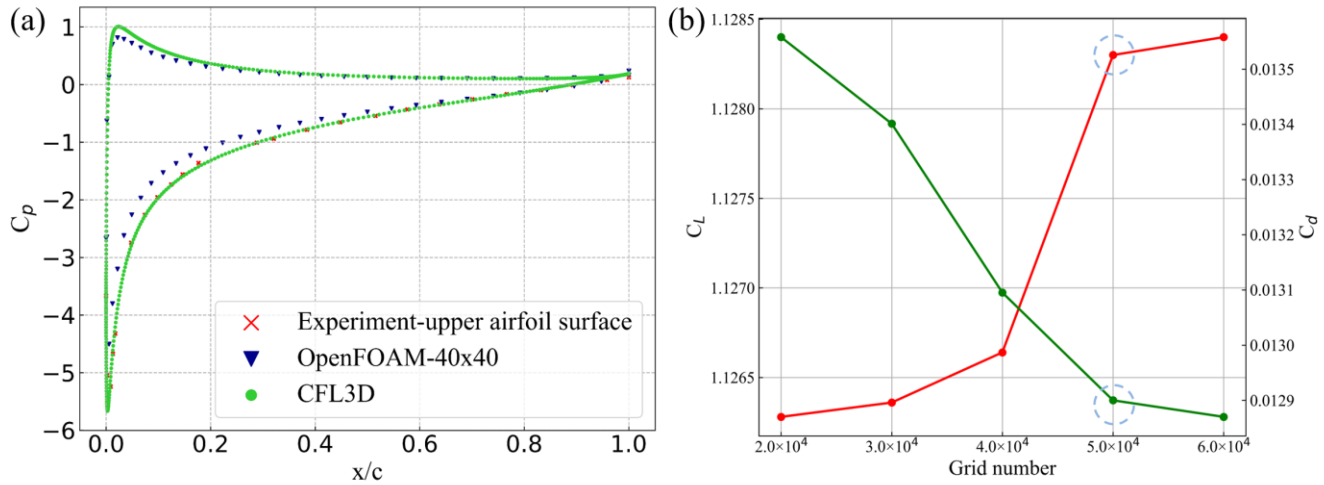

Figure 6. (**a**) Surface pressure coefficient from different data sources vs. normalized chord length; (**b**) grid independence study.

2.3.2. Airfoil Representation

The traditional parameterization techniques, such as basic profiles, geometric parameterized variables, were widely used in conventional aerodynamic surrogate models such as polynomial regression, Kriging and RBF (Radial Basis Function) regression network, nevertheless, it is not effective for CNN since convolutional operations on parametric vectors only move along one direction. The binary matrices to parameterize airfoils can work with CNN but they contain less information, i.e., 0 and 1, which cannot reflect the distance from the given point (especially the point near the airfoil) to the airfoil profile, leading the neural network model for point-to-point flow field prediction not being trained efficiently. To capture the details of the airfoil geometry and to classify grid points into a learnable format [36,50], SDF is used in this work to obtain a grid sampling of the minimum distance to the airfoil surface, where a constant grid size of 208 × 340 is adopted. SDF is widely used in applications of rendering, segmentation, and extracting structural information of different shapes. It offers a universal parameterization of various geometry shapes and works efficiently with neural networks. To speed up the generation of SDF parameters, the fast marching method [51] is adopted to enhance the boundary information of airfoil

geometries for neural networks. As a result, the mathematical definition of SDF used in the paper is given as follows:

$$S(i) = \begin{cases} d(i, \partial\Omega) & i \notin \Omega \\ 0 & i \in \partial\Omega \\ -d(i, \partial\Omega) & i \in \Omega \end{cases} \tag{22}$$

where Ω is the close area covered by airfoil, $\partial\Omega$ refers to its boundary, i stands for a given pixel point, and $d(i, \partial\Omega)$ is computed as

$$d(i, \partial\Omega) = \min_{I \in \partial\Omega}(|i - I|) \tag{23}$$

since I could be any point on the boundary wall, $d(i, \partial\Omega)$ by Equation (23) measures the shortest distance from point i to the airfoil boundary.

Figure 7 shows the SDF representation of airfoil NACA8412 at AOA equals to 1°. It could be seen that SDF takes negative values on points inside the airfoil, while it takes positive values on points outside the airfoil and increases in directions pointing outwards to the airfoil boundary. The value of SDF contains both local geometry details and auxiliary information of the global geometry structure, which promotes the training efficiency of the neural network.

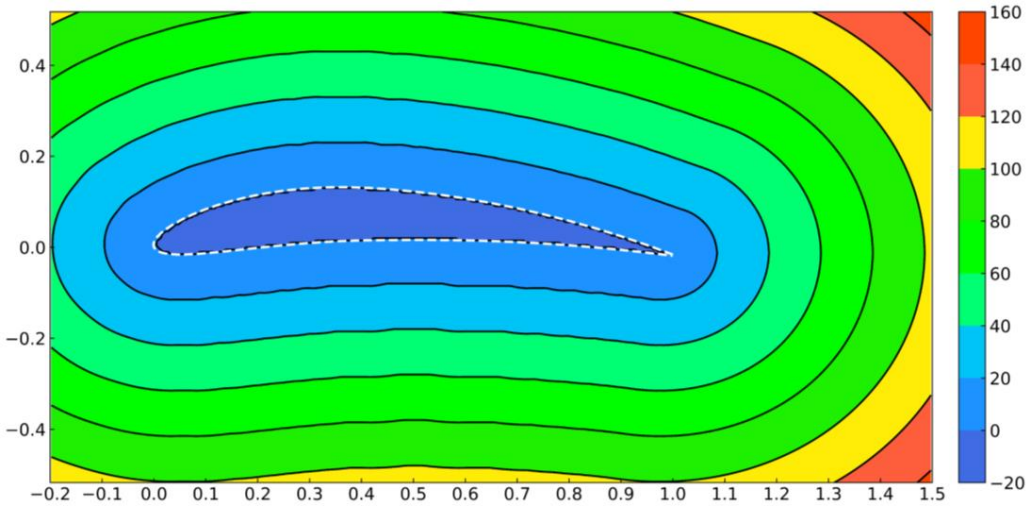

Figure 7. The SDF contour for NACA8412 airfoil on a 208 × 340 Cartesian grid at AOA = 1°. The airfoil boundary is the dashed line in white color.

2.3.3. Data Preparation and Processing

Training of the neural network generally requires a pre-existed dataset with a sufficiently large number of known pairs of input and output. In this paper, as output (pressure and velocity fields) in the dataset is the CFD result of the given input, a number of cases are built up by changing parameters of inputs in three dimensions: the airfoil geometry, AOA, and Reynolds number. Change of the airfoil geometry is conducted by modifying the maximum camber and its position along the chord, during which maximum thickness is kept at a constant value, as usually required by the aerodynamic design of the airfoil. In this manner, 82 airfoil profiles are generated based on the original airfoil NACA0012, which is given in Figure 8, along with 30 examples of modified profiles. Then the SDF representations of these airfoil geometries are obtained by the method given in Section 2.3.2. For each parameterized airfoil, 7 AOAs and 3 Reynolds numbers shall be considered, from the array of [0°, 1°, 2°, 3°, 4°, 5°, 6°] and [3×10^6, 5×10^6, 7×10^6], respectively. In practice, the value of Reynold number, after being divided by 1×10^6, is merged into SDF to capture the enough mapping from the same airfoil geometry. As a result, the CFD dataset used in this work contains 1722 (82 × 8 × 3) pairs of input and corresponding CFD output.

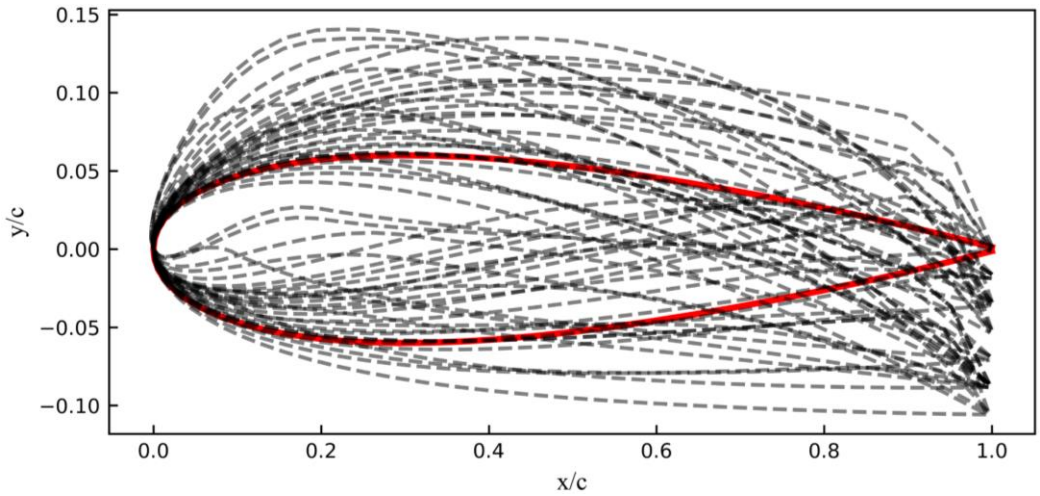

Figure 8. The geometry of 30 airfoils sampled from the airfoil library (NACA0012 highlighted by the solid red line).

As the majority of the flow variations take place within a close region near the airfoil surface, we focus the training and prediction process of CNN-DCNN on a selected box area as the inference region to reduce memory usage, instead of working with the entire CFD domain covered by computational grid. Concentrated on the near wall region, the rectangular box area ranges from $-0.2c$ to $1.5c$ in x direction, and $-0.52c$ to $0.52c$ in y direction, where the coordinates x and y are defined in the way as those in Figure 8. The CFD data of pressure and velocity is then extracted from the inference region and interpolated into a matrix with a size of 208×340. Thus, stacking SDF and the flow field data, the training input of the CNN-DCNN model is designed as

$$Input = [S, P, U] \tag{24}$$

where S, P, and U are all normalized to promote the training efficiency of the network model. Moreover, it should be pointed out that S, P, and U are all masked with a binary matrix which is filled with 0 and 1 for points inside and outside of the given airfoil, respectively, since there is no need to pay attention to the inside of the airfoil, where there is no flow field.

3. Results and Discussion

3.1. Effect of Training Parameters

To study the effect of hyperparameters on the convergence and accuracy of the proposed CNN-DCNN model during the training process, different sets of values are tested for the training parameters, namely the learning rate, the mini-batch size and the random deactivation rate (dropout). The parameters are studied by means of the control variable, so a large number of parameter combinations have to be tested. Results with parameter values around the finally chosen ones are reported in Figure 9, where lines in red color stand for the reference case, in which the learning rate, the mini-batch size, and the random deactivation rate are set to 0.005, 32, and 0.2, respectively, as the reference values. Throughout the training process, results are sampled every 200 epochs. It should be noted that the sampling frequency of 200 just denotes the frequency of recording model accuracy, which is set to save time merely and will not affect the training of model.

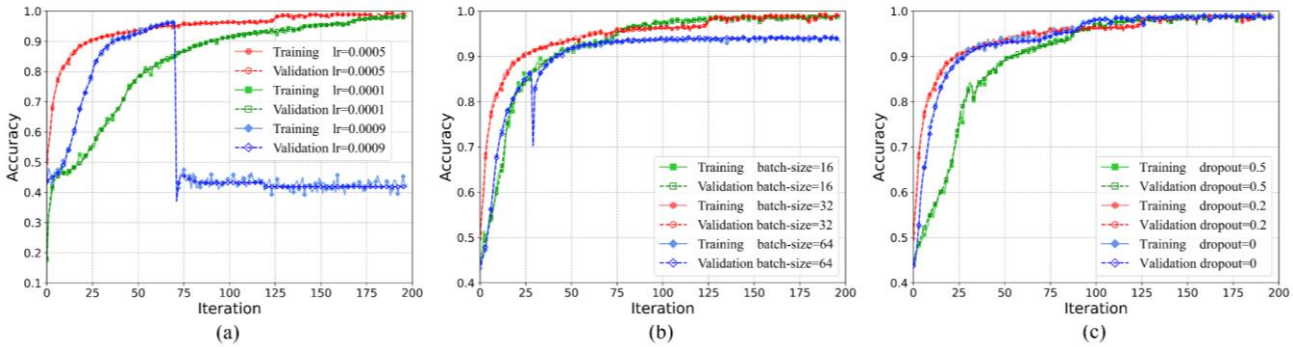

Figure 9. Convergence history (sampled every 200 epochs) under different hyperparameters of the (**a**) learning rate, (**b**) batch-size, (**c**) random deactivation rate (dropout).

Figure 9a shows the convergence curves with values of the learning rate (abbreviated as "lr" in the figure), for both the training and validation accuracy of the proposed model. Generally, the validation accuracy fluctuates closely around the training accuracy, and both curves converge normally with the a learning rate of 0.0005. Smaller learning rate of 0.0001 would slow down the convergence, although the same level of accuracy could be achieved with enough number of iterations. Larger learning rate of 0.0009 does accelerate the learning process in the early stage, however, it would be trapped by a local minimum which failed the training process.

Impacts of mini-batch size and dropout are displayed respectively in panels (b) and (c) of Figure 9. Compared with the reference value of 32, the smaller mini-batch size turns out to have higher gradient oscillation between adjacent iterations, which is not conducive to the convergence of the model. A larger mini-batch size, however, would get caught by a local optimum, and thus could not reach the same level of accuracy as the other two setups. As for the choice of dropout, which is employed by the CNN-DCNN model to avoid overfitting, Figure 9c implies that the appropriate value (0.2 in the current case) could accelerate the convergence of the training process, and in turn benefit the generalization ability of the model handling unseen data during the testing stage. The hyperparameters adopted for training and prediction in this work are summarized in Table 3.

Table 3. Hyperparameters setting.

Hyperparameter	Learning Rate	Mini-Batch Size	Dropout	Regularization Coefficient
Notation	α	k	d	λ
Value	0.0005	32	0.2	0.0001

3.2. Flow Field Prediction Analysis

Following the methodology given in Section 2.2, the training datasets of the CFD results are fed to the CNN-DCNN model, with training parameters detailed in Section 3.1. Prediction of flow field of the trained model is then compared with the CFD results of the test datasets. Table 4 shows the mean accuracy of prediction for both pressure and velocity fields. Although both prediction accuracies are above 98%, the accuracy for velocity prediction is slightly better than that of the pressure. This is due to the non-negligible numerical differences between the pressure and velocity fields, despite being normalized before being fed into the neural network. Given the accuracy level of the model prediction, it is worth mentioning that prediction for a single case takes only 25 ms on a graphics processing unit (GPU), while a corresponding CFD calculation takes 3 min. Admittedly, the training process would spend 25 min before the proposed model become functional, as shown in Table 5, the CNN-DCNN framework might still be able to greatly reduce the computational cost of a pure CFD-based method, especially for aerodynamic design and optimization where a large number of parameters have to be studied and tested.

Table 4. Mean accuracy of the physical field prediction for the test data.

Field	Pressure Field	Velocity Field
Accuracy (%)	98.2302	99.6016

Table 5. Comparison of time cost for the physical field prediction by the CNN-DCNN model and CFD simulation. The GPU is RTX 3080.

Method	Data Preparation	Train Cost	Prediction Cost
CFD	86.1 h	-	3 min
CNN-DCNN (GPU)	-	25 min	24 ms

Figure 10 compares the predicted and calculated flow fields of 3 airfoils with different geometries and AOAs, at the same Reynolds number of 5×10^6. Visual inspections indicate that the predicted fields are almost identical to calculated ones for both pressure and velocity, with small relative errors for the majority part of the flow region. The predicted stagnation point of the leading edge is clearly visible, where the positive pressure reaches the maximum value and the airflow velocity drops to zero. For both predicted and calculated flow fields, we could find a local high-velocity region on the upper surface of the airfoil, as well as a long wake region behind the trailing edge. Typically, the maximum relative errors are found around the leading and trailing edges of the airfoil, due to the flow separation induced by the circulation and incoming flow. Specifically, the error band of velocity on the trailing edge of the airfoil almost coincide with the wake caused by flow separation. Therefore, the reasons for these errors may be the sharp numerical changes, but it should be noted that the errors appear much smaller. Comparison of the predicted and calculated field indicates that the proposed CNN-DCNN model is accurately able to predict the physical fields for the various airfoils, thus potentially substituting part of CFD workloads during aerodynamic design and optimization.

Figure 11 compares of the gradient of the horizontal component of velocity post-processed from the calculated and predicted velocity field, which is closely coupled with viscous force, especially in the near wall region. As we can see from the relative error, the velocity gradient of the predicted field matches well with CFD calculation. The drag coefficient is one of the most widely used performance indicators for airfoil design, which measures the integral effect of pressure and velocity field around the airfoil. Table 6 gives the calculated and drag coefficient of the two cases in Figure 11, where the relative errors of the model prediction are below 1.3% compared with corresponding CFD calculations.

Table 6. Drag coefficients for two cases in Figure 11.

Case	True (CFD)	Prediction (CNN-DCNN)	Relative Error (%)
(a)	0.00908	0.00898	1.1%
(b)	0.00394	0.00389	1.3%

To further illustrate the capability of the proposed model to capture different input features, the CFD calculated and model predicted fields are compared under different AOAs. Figure 12 shows the pressure and velocity fields for airfoil NACA9812 at the same Reynolds number with AOAs of 0°, 4° and 6°. We are not surprised to see that the predicted fields correspond closely to the calculated ones in both Figures, in which the error bands of velocity fields almost coincide with the wake. Reproduction of the calculated fields indicates that the proposed model is able to take advantage of the parameterized information within the input matrix and figure out the mapping between different element features and the input flow field, benefiting from the SDF representation of geometry features.

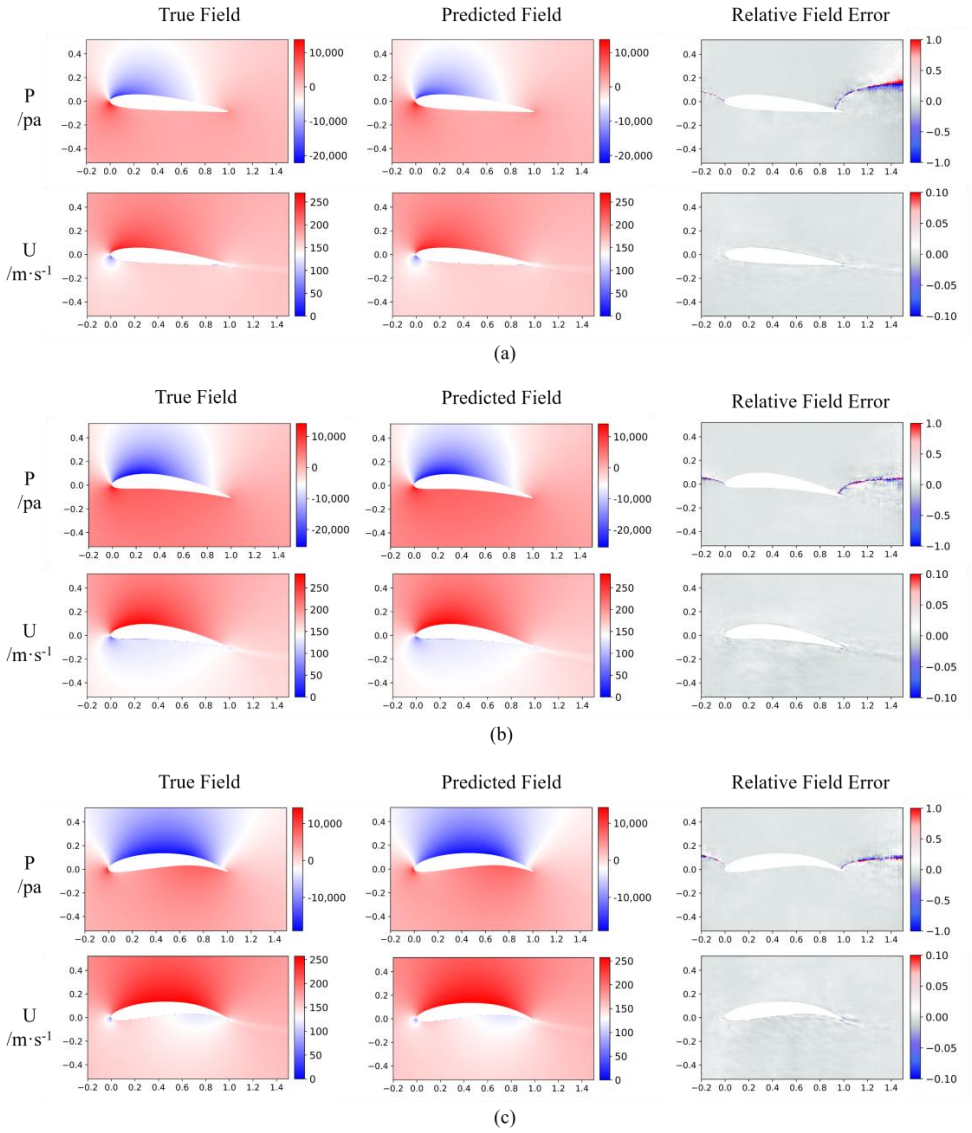

Figure 10. The true (CFD) and predicted physical fields for three airfoils under $Re = 5 \times 10^6$: (**a**) NACA2412 with 5° AOA; (**b**) NACA7412 with 6° AOA; (**c**) NACA9612 with 1° AOA.

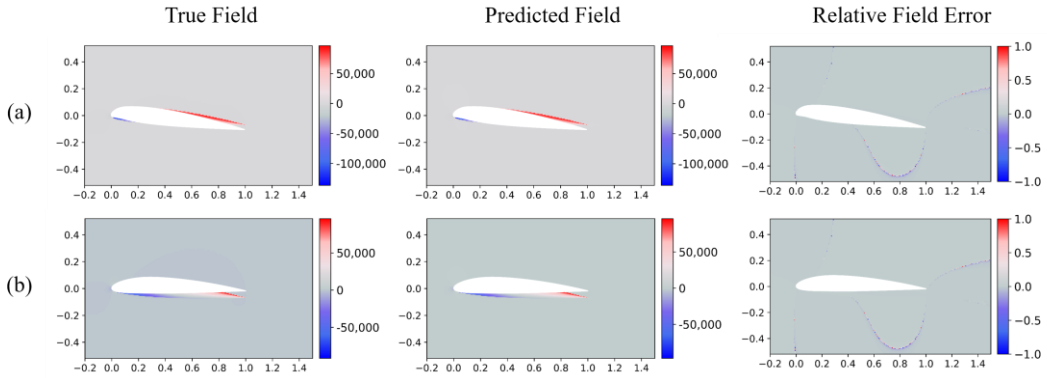

Figure 11. The true (CFD) and predicted velocity gradient fields for three airfoils under $Re = 7 \times 10^6$: (**a**) NACA3112 with 6° AOA and (**b**) NACA3312 with 1° AOA.

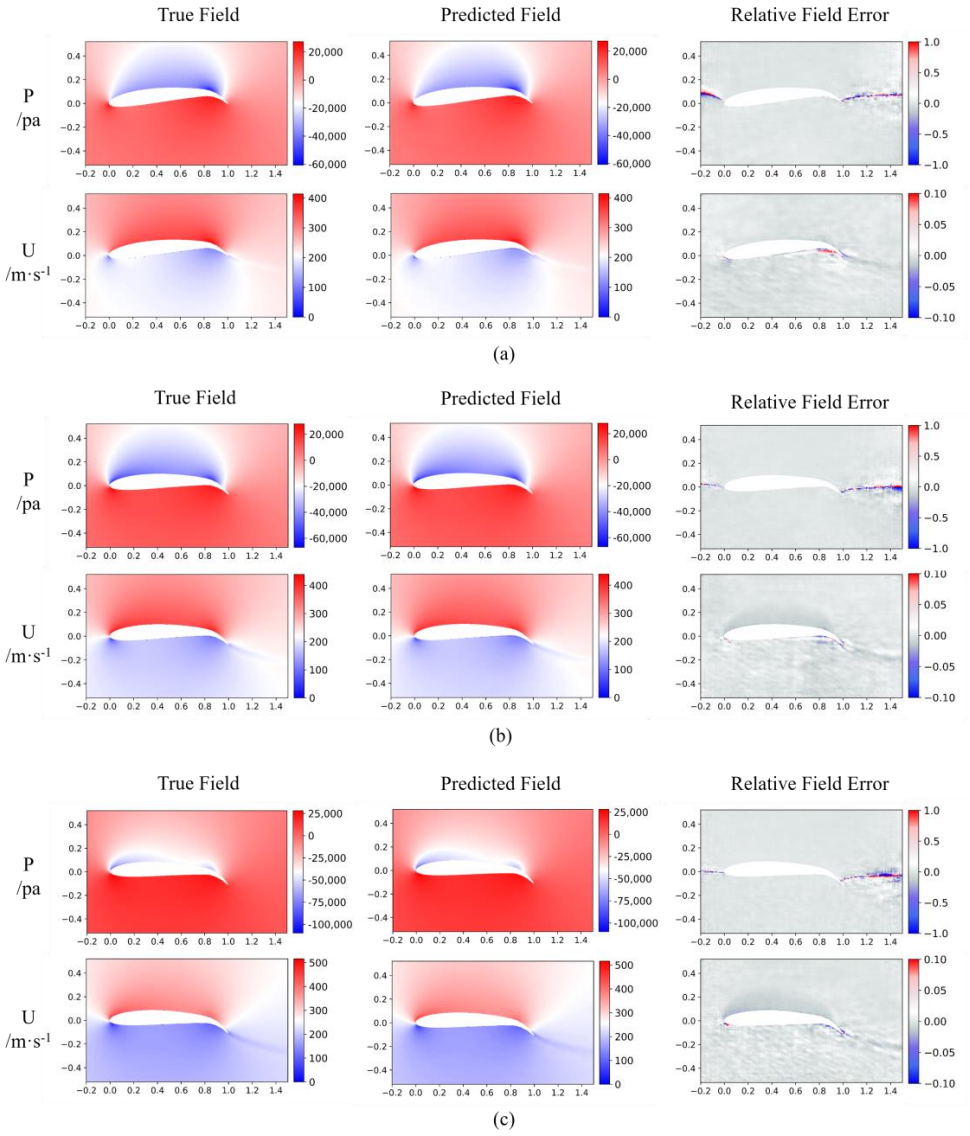

Figure 12. The true (CFD) and predicted physical fields for NACA9812 airfoil under $Re = 7 \times 10^6$ across three AOAs: (**a**) 0°; (**b**) 4°; (**c**) 6°.

A more rigorous evaluation of the prediction quality is provided in Figure 13, where the contour lines of flow fields are directly compared, with different airfoil geometries, AOA, and Reynolds numbers. We could find out that the contours of the flow fields predicted by the CNN-DCNN model are consistent, if not coincident with those from the CFD simulations. The good match around the flow separation regions indicate that the proposed model has the potential to deal with highly non-linear problems.

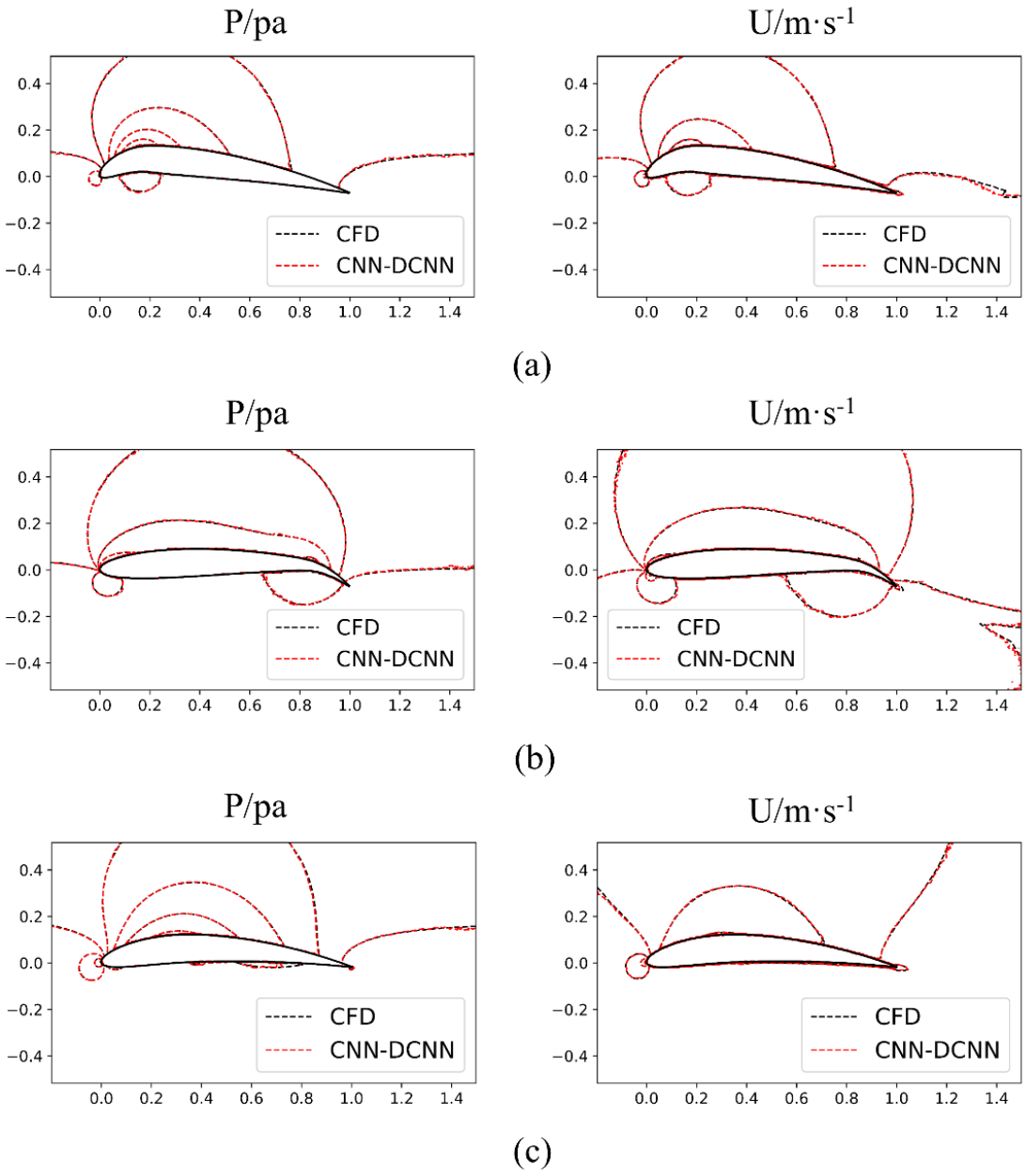

Figure 13. Comparison of the pressure and velocity contours for three cases: (**a**) NACA9212, AOA = 4°, $Re = 3 \times 10^6$; (**b**) NACA8812, AOA = 4°, $Re = 5 \times 10^6$; (**c**) NACA7412, AOA = 1°, $Re = 7 \times 10^6$.

3.3. Comparison with POD

For a more quantitative study on the interpretability of our CNN-DCNN model, we use the method of POD to investigate the reconstructed physical fields. POD extracts essential characteristics of discrete data by describing multidimensional stochastic processes with low-dimensional approximations. Its basic idea is to decompose the dynamic system into a set of basic functions (modes), where the principle of determining the basis function is to make the lowest order modes retain as much "energy" as possible from the original high-order system. Considering a given pressure or velocity fields as a two-dimensional matrix $\mathbf{A}_{m\times n}$ (where $m \times n$ is 208 × 340 in this work), decomposition by POD gives:

$$\mathbf{A}_{m\times n} = \mathbf{W}_{m\times m} \sum_{m\times n} \mathbf{Z}^*_{n\times n} \tag{25}$$

where the column vector of $\mathbf{W}_{m\times m}$ is the spatial modes of $\mathbf{A}_{m\times n}$, the row vector of $\mathbf{Z}^*_{n\times n}$ represents the temporal modes and $\sum_{m\times n}$ is the eigenvalue matrix. $\mathbf{A}_{m\times n}$ can be further described as follows:

$$\mathbf{A}_{m\times n} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n] \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \ldots & & \\ & & & \lambda_r & \\ & & & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \ldots \\ \mathbf{z}_n \end{bmatrix} = \sum_{j=1}^{r} \lambda_j \mathbf{w}_j \mathbf{z}_j \tag{26}$$

where λ_j is the non-zero eigenvalue of $\mathbf{A}_{m\times n}$, $\mathbf{w}_j$ is the column vector of $\mathbf{W}_{m\times n}$, $\mathbf{z}_j$ is he row vector of $\mathbf{Z}^*_{n\times n}$ and $\lambda_j \mathbf{w}_j \mathbf{z}_j$ represents the jth mode of $\mathbf{A}_{m\times n}$. POD decomposition is "optimal" by its mathematical nature: the first mode ($\lambda_1 \mathbf{w}_1 \mathbf{z}_1$) contains more energy than the second mode ($\lambda_2 \mathbf{w}_2 \mathbf{z}_2$), $\lambda_2 \mathbf{w}_2 \mathbf{z}_2$ is more energetic than $\lambda_3 \mathbf{w}_3 \mathbf{z}_3$, which is again more energetic than $\lambda_4 \mathbf{w}_4 \mathbf{z}_4 \ldots$ Consequently, reconstruction of the original $\mathbf{A}_{m\times n}$ can be approximated by retaining only the leading modes in Equation (26), the number of the retained modes depend on the acceptable error threshold.

After applying POD decomposition to the pressure and velocity fields from both CFD calculation and model prediction, the first thing we try to compare is the eigenvalue, as it is a direct measurement of the energy contained in the corresponding POD mode. Figure 14 gives the eigenvalues of 3 case setups of airfoil geometry, AOA and Reynolds numbers, which would be referred to as case 1, case 2 and case 3. For all three cases, the energy cascade is clearly visible from the monotonic decrease of eigenvalues, and model prediction agrees almost exactly with the original CFD calculation for the leading 20 to 30 modes. For the trailing POD modes, however, eigenvalues by the CNN-DCNN model tend to be higher than CFD by an order of magnitude. Considering the optimality characteristic of the POD method, the deviations in the high order range have litter impact on the overall accuracy of prediction, as the low order modes already extract almost the entire system energy from the original flow fields. In this sense, if we are looking at the global effect considering all the POD modes, error of eigenvalue by the CNN-DCNN model is typically lower than 1%, as given in Table 7.

Table 7. Global error of eigenvalue of the CNN-DCNN model for cases in Figure 14.

Relative Error	Pressure Field (%)	Velocity Field (%)
Case 1	0.59	0.12
Case 2	0.44	0.16
Case 3	0.29	0.12

As eigenvalue provides information in terms of the energy contained by a given POD mode, the intrinsic characteristic of the fluid dynamics is only visible from the distribution of POD mode itself, or more intuitively, the reconstructed flow field by leading POD modes according to Equation (26). Taking case 3 as an example, Figure 15 displays the reconstructed flow field with different numbers of leading POD modes, as compared with the predicted results of the CNN-DCNN model. It could be found that to have a comparable level of resemblance to the original (CFD) flow field as CNN-DCNN model prediction, POD reconstruction requires at least 20 to 30 leading modes for the pressure field, and about 50 for the velocity fields. This observation is confirmed by the comparison of accuracies of different level POD reconstructions summarized in Table 8: The reconstructed field could not be as accurate as CNN-DCNN until more than about 50 POD modes are retained, beyond this value, reconstruction is more accurate but the improvement is quite limited as the accuracy is already approaching 100%. Besides, it is worth pointing out that POD by itself is typically working with a dataset that is already available (for modal decomposition and identification [52,53]) rather than a prediction tool such as the proposed CNN-DCNN model, which after trained, is expected to directly provide flow field data that is not necessarily pre-existed. Consequently,

as predicted flow fields correlate closely to the leading POD modes that contain almost the entire energy (eigenvalues) and dynamic information (eigen functions), it is reasonable to be optimistic in terms of the capability of the CNN-DCNN model to retain at least part of the intrinsic information of the flow dynamics.

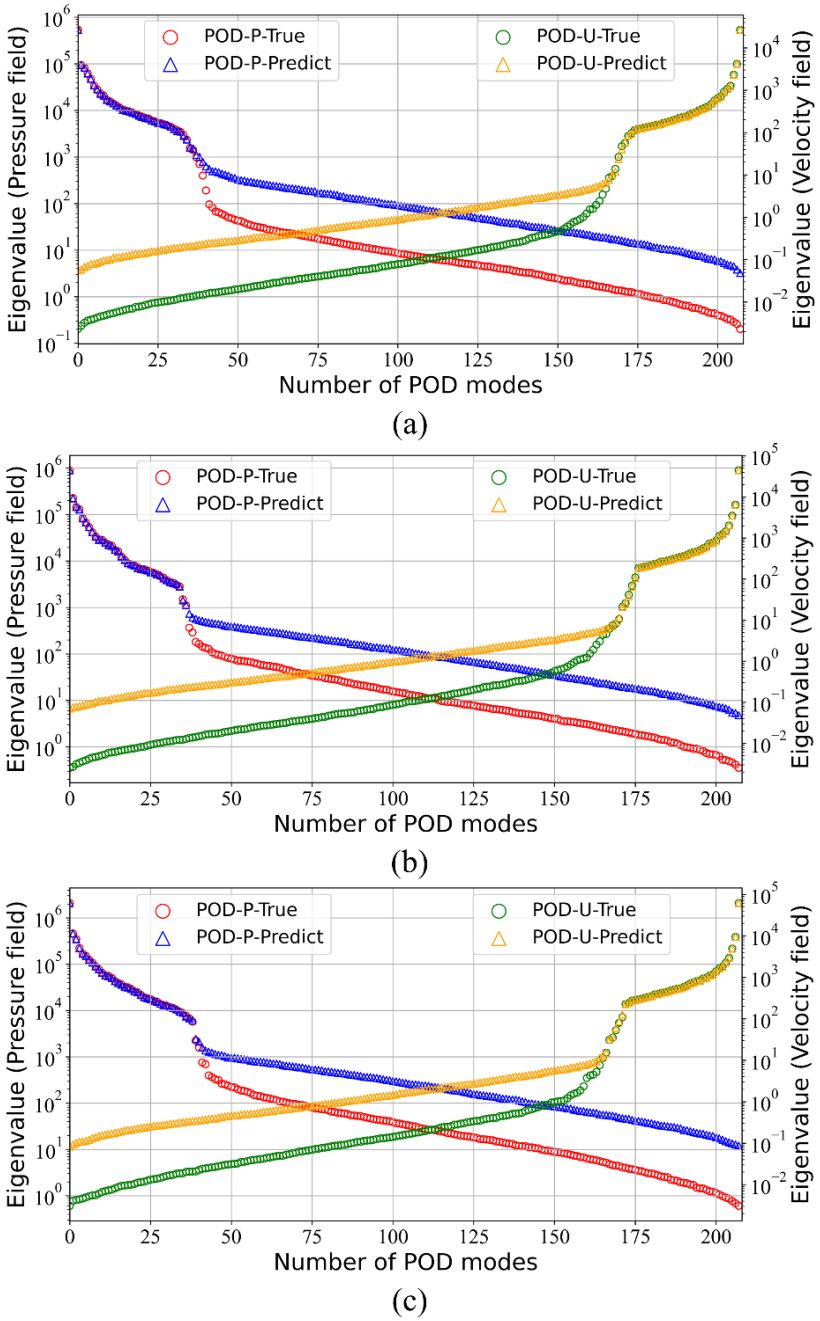

Figure 14. POD eigenvalues calculated from the true and predicted flow fields CFD and CNN-DCNN model prediction: (**a**) case 1: NACA9712, AOA = 0°, $Re = 3 \times 10^6$; (**b**) case 2: NACA2412, AOA = 5°, $Re = 5 \times 10^6$; (**c**) case 3: NACA6212, AOA = 3°, $Re = 7 \times 10^6$.

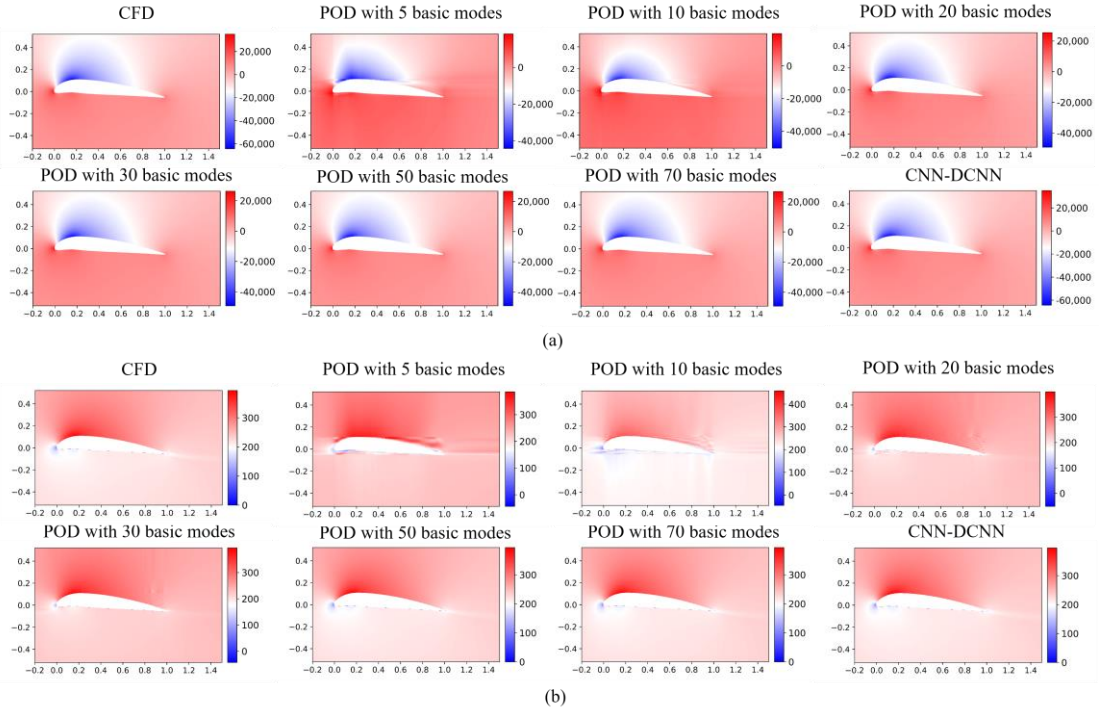

Figure 15. POD/CNN-DCNN comparison of the reconstructed physical field (case 3: NACA6212, AOA = 3°, $Re = 7 \times 10^6$): (**a**) pressure field; (**b**) velocity field.

Table 8. CNN-DCNN/POD comparison of accuracy for case 3.

Method	Pressure Field (%)	Velocity Field (%)
CNN-DCNN	99.1913	99.7574
POD with 5 basic modes	86.4541	93.2564
POD with 10 basic modes	92.8026	95.6101
POD with 20 basic modes	96.9733	97.6766
POD with 30 basic modes	98.6831	98.9191
POD with 50 basic modes	99.9648	99.9925
POD with 70 basic modes	99.9825	99.9969

3.4. Investigation of the Extensionality of the CNN-DCNN Model

All the previous results and analyses are based on flow conditions with fixed Reynolds numbers (3×10^6, 5×10^6 or 7×10^6), as the far-field velocity of airfoil changes continuously in practice, we study the extensionality of the CNN-DCNN model with unseen Reynolds numbers not available in the training datasets. Specifically, the input parameters of the three cases analyzed in Section 3.3 are reused here with modified Reynolds numbers, namely $Re = 2.5 \times 10^6$ (case 4), $Re = 6 \times 10^6$ (case 5) and $Re = 8 \times 10^6$ (case 6). Compared with the training dataset, case 4 is an interpolated case while the other two are extrapolated cases. Figure 16 shows the predicted flow fields of the three extended cases, in comparison with those by CFD calculation. It is appreciated that the model predictions are quite close to the results obtained by much more expensive CFD calculations. According to the relative errors displayed in the last column of Figure 16, the largest deviation appears in case 4, as the NACA9712 airfoil has the largest camber and its position along the chord is closer to the trailing edge compared with the other two cases. The prediction accuracies for the three cases are reported in Table 9, which confirm that airfoils with larger deformation turn to decreasing the prediction accuracy of the CNN-DCNN model, possibly due to more complex flow structures.

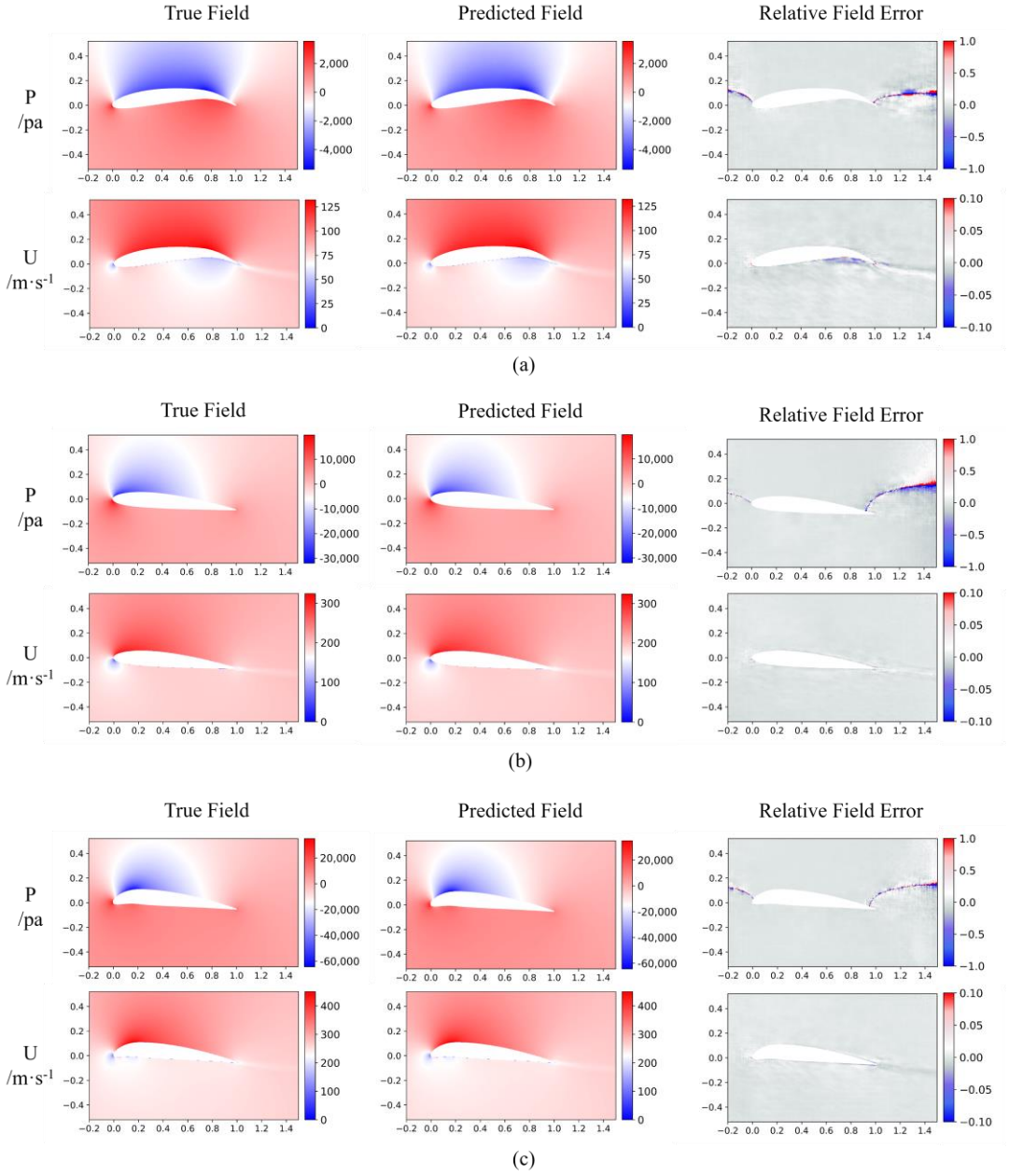

Figure 16. The true (CFD) and predicted flow fields of three cases under the extended Reynolds numbers: (**a**) case 4: NACA9712, AOA = 0°, $Re = 2.5 \times 10^6$; (**b**) case 5: NACA2412, AOA = 5°, $Re = 6 \times 10^6$; (**c**) case 6: NACA6212, AOA = 3°, $Re = 8 \times 10^6$.

Table 9. Accuracy for three cases under the extended Reynolds numbers.

Accuracy	Pressure Field (%)	Velocity Field (%)
Case 4 ($Re = 2.5 \times 10^6$)	98.6469	99.5619
Case 5 ($Re = 6 \times 10^6$)	98.8817	99.7842
Case 6 ($Re = 8 \times 10^6$)	99.1802	99.7544

Following the routine of Section 3.2, contour lines of pressure and velocity fields of the extended cases are explicitly given in Figure 17, so that discrepancies in spatial flow structures are more visible. For both pressure and velocity, the CNN-DCNN model managed to reproduce contour lines with good accuracy even in the separation area,

suggesting good extensionality of the proposed model when dealing with unseen flow situations, at least in the investigated range of Reynolds numbers.

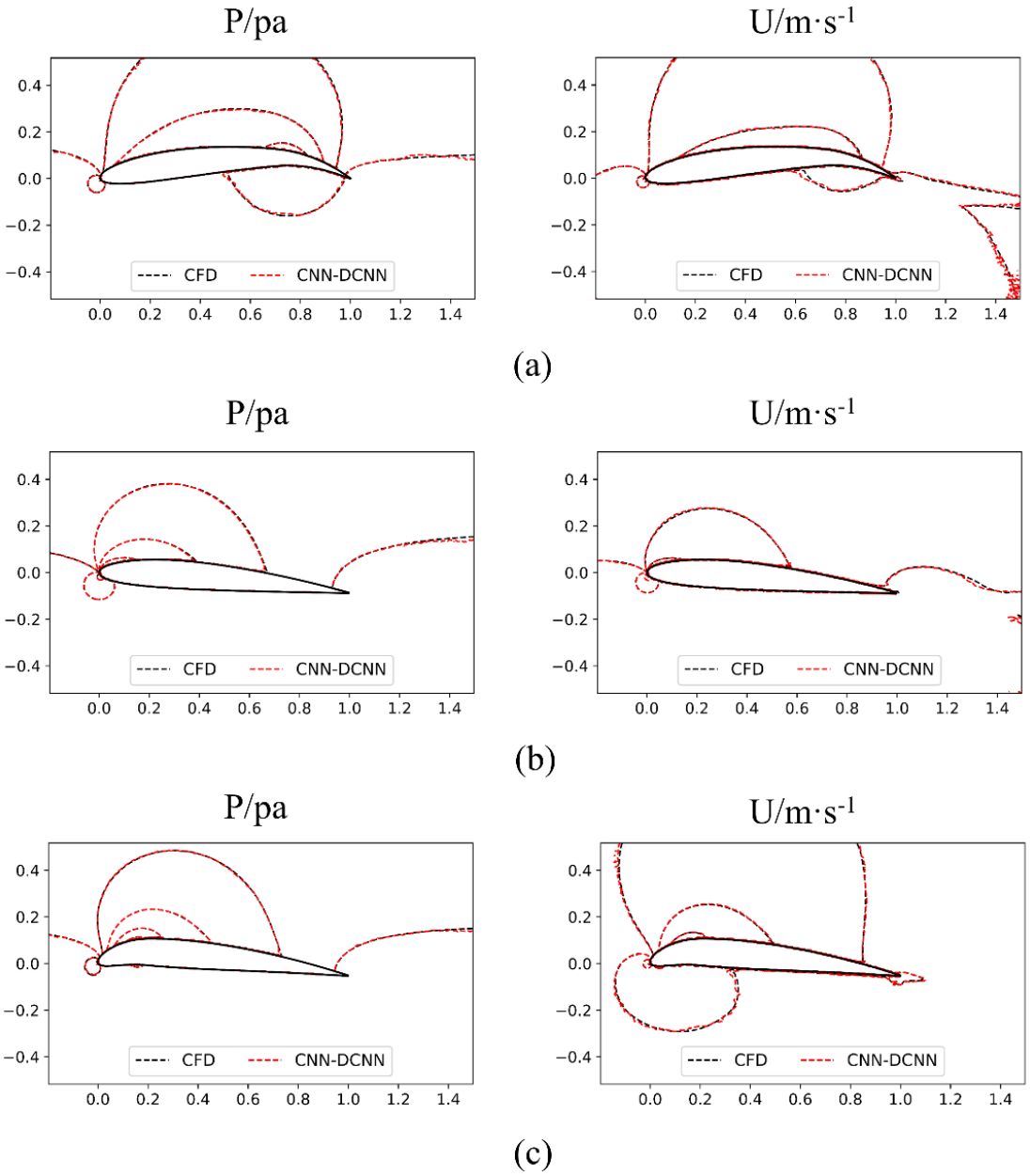

Figure 17. Comparison of the pressure and velocity contours for three cases: (**a**) case 4: NACA9712, AOA = 0°, $Re = 2.5 \times 10^6$; (**b**) case 5: NACA2412, AOA = 5°, $Re = 6 \times 10^6$; (**c**) case 6: NACA6212, AOA = 3°, $Re = 8 \times 10^6$.

4. Conclusions

In this work, we proposed a data-driven reduced-order model based on the CNN-DCNN framework to predict the flow field over a series of airfoils. The effects of training parameters are studied, then the model prediction results are compared with those obtained by CFD calculation, to evaluate the performance, accuracy, and extensionality of the proposed CNN-DCNN model, as well as its capability to capture the intrinsic characteristics of the flow dynamics. The following conclusions can be summarized:

(1) Using SDF representation of airfoil geometry, with Reynolds number as an auxiliary element fed into the neural network, the trained CNN-DCNN model is able to learn

from the implicit constitutive relation of Reynolds stress and accurately map the geometries of the airfoils to the corresponding flow field.

(2) The CNN-DCNN model shows the remarkable predicting ability for the flow field over NACA0012-series airfoils with prediction accuracy up to 98.23% for the pressure field and 99.60% for velocity field.

(3) The prediction process of CNN-DCNN model is faster than a corresponding CFD calculation by three orders of magnitude, which leaves broad prospects in the application of aerodynamic design and optimization, especially considering the good extensionality of the model working with flow conditions with unseen Reynolds numbers.

(4) The CNN-DCNN model can capture the spatial structure of the flow field, achieving accurate prediction results around the flow separation region, comparison with reconstructed field by POD method indicates that it is able to retain the essential information of coherent structures from highly non-linear flow systems.

Author Contributions: Conceptualization, M.-Y.W. and W.-T.W.; methodology, M.-Y.W. and Y.W.; validation, X.-Y.Y.; writing—original draft preparation, M.-Y.W.; writing—review and editing, Y.W. and Z.-H.C.; supervision, M.-Y.W. and N.A.; All authors have read and agreed to the published version of the manuscript. M.-Y.W. and Y.W. contributed to this paper equally.

Funding: This research was funded by the Natural Science Foundation of Jiangsu Province (Grant No. BK20201302, Grant No. BK20220954) and the Fundamental Research Funds for the Central Universities (No. 30919011401).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, T.-T.; Wang, Z.-G.; Huang, W.; Yan, L. A review of parametric approaches specific to aerodynamic design process. *Acta Astronaut.* **2018**, *145*, 319–331. [CrossRef]
2. Arunvinthan, S.; Raatan, V.S.; Pillai, S.N.; Pasha, A.A.; Rahman, M.M.; Juhany, K.A. Aerodynamic characteristics of shark scale-based vortex generators upon symmetrical airfoil. *Energies* **2021**, *14*, 1808. [CrossRef]
3. Raheem, M.A.; Edi, P.; Pasha, A.A.; Rahman, M.M.; Juhany, K.A. Numerical study of variable camber continuous trailing edge flap at off-design conditions. *Energies* **2019**, *12*, 3185. [CrossRef]
4. Eldred, M.; Dunlavy, D. Formulations for surrogate-based optimization with data fit, multifidelity, and reduced-order models. In Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA, USA, 6–8 September 2006; p. 7117.
5. Hill, W.J.; Hunter, W.G. A review of response surface methodology: A literature survey. *Technometrics* **1966**, *8*, 571–590. [CrossRef]
6. Hacioglu, A. Fast evolutionary algorithm for airfoil design via neural network. *AIAA J.* **2007**, *45*, 2196–2203. [CrossRef]
7. Rai, M.M.; Madavan, N.K. Aerodynamic design using neural networks. *AIAA J.* **2000**, *38*, 173–182. [CrossRef]
8. Mullur, A.A.; Messac, A. Extended radial basis functions: More flexible and effective metamodeling. *AIAA J.* **2005**, *43*, 1306–1315. [CrossRef]
9. Clarke, S.M.; Griebsch, J.H.; Simpson, T.W. Analysis of support vector regression for approximation of complex engineering analyses. *J. Mech. Des.* **2005**, *127*, 1077–1087. [CrossRef]
10. Han, Z.-H.; Görtz, S. Hierarchical kriging model for variable-fidelity surrogate modeling. *AIAA J.* **2012**, *50*, 1885–1896. [CrossRef]
11. Laurenceau, J.; Sagaut, P. Building efficient response surfaces of aerodynamic functions with kriging and cokriging. *AIAA J.* **2008**, *46*, 498–507. [CrossRef]
12. Qiu, Y.; Bai, J. Stationary flow fields prediction of variable physical domain based on proper orthogonal decomposition and kriging surrogate model. *Chin. J. Aeronaut.* **2015**, *28*, 44–56. [CrossRef]
13. Liu, X.; Zhu, Q.; Lu, H. Modeling multiresponse surfaces for airfoil design with multiple-output-Gaussian-process regression. *J. Aircr.* **2014**, *51*, 740–747. [CrossRef]
14. Chiplunkar, A.; Bosco, E.; Morlier, J. Gaussian process for aerodynamic pressures prediction in fast fluid structure interaction simulations. In *World Congress of Structural and Multidisciplinary Optimisation*; Springer: Cham, Switzerland, 2017; pp. 221–233.
15. Santos, M.; Mattos, B.; Girardi, R. Aerodynamic coefficient prediction of airfoils using neural networks. In Proceedings of the 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 7–10 January 2008; p. 887.

16. Wallach, R.; Mattos, B.; Girardi, R.; Curvo, M. Aerodynamic coefficient prediction of transport aircraft using neural network. In Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 9–12 January 2006; p. 658.
17. Wu, X.; Zhang, W.; Peng, X.; Wang, Z. Benchmark aerodynamic shape optimization with the POD-based CST airfoil parametric method. *Aerosp. Sci. Technol.* **2019**, *84*, 632–640. [CrossRef]
18. Raul, V.; Leifsson, L. Surrogate-based aerodynamic shape optimization for delaying airfoil dynamic stall using Kriging regression and infill criteria. *Aerosp. Sci. Technol.* **2021**, *111*, 106555. [CrossRef]
19. Zhu, L.; Zhang, W.; Kou, J.; Liu, Y. Machine learning methods for turbulence modeling in subsonic flows around airfoils. *Phys. Fluids* **2019**, *31*, 015105. [CrossRef]
20. Shan, S.; Wang, G.G. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Struct. Multidiscip. Optim.* **2010**, *41*, 219–241. [CrossRef]
21. Kou, J.; Zhang, W. A hybrid reduced-order framework for complex aeroelastic simulations. *Aerosp. Sci. Technol.* **2019**, *84*, 880–894. [CrossRef]
22. Lee, S.; Ha, J.; Zokhirova, M.; Moon, H.; Lee, J. Background information of deep learning for structural engineering. *Arch. Comput. Methods Eng.* **2018**, *25*, 121–129. [CrossRef]
23. Kutz, J.N. Deep learning in fluid dynamics. *J. Fluid Mech.* **2017**, *814*, 1–4. [CrossRef]
24. Hasegawa, K.; Fukami, K.; Murata, T.; Fukagata, K. CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers. *Fluid Dyn. Res.* **2020**, *52*, 065501. [CrossRef]
25. Lee, S.; You, D. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *J. Fluid Mech.* **2019**, *879*, 217–254. [CrossRef]
26. Han, R.; Wang, Y.; Zhang, Y.; Chen, G. A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network. *Phys. Fluids* **2019**, *31*, 127101.
27. Wang, J.; He, C.; Li, R.; Chen, H.; Zhai, C.; Zhang, M. Flow field prediction of supercritical airfoils via variational autoencoder based deep learning framework. *Phys. Fluids* **2021**, *33*, 086108. [CrossRef]
28. Renganathan, S.A.; Maulik, R.; Rao, V. Machine learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil. *Phys. Fluids* **2020**, *32*, 047110. [CrossRef]
29. Yan, Z.; Yang, H.; Li, F.; Lin, Y. A Deep Learning Approach for Short-Term Airport Traffic Flow Prediction. *Aerospace* **2021**, *9*, 11. [CrossRef]
30. Yu, J.; Hesthaven, J.S. Flowfield reconstruction method using artificial neural network. *AIAA J.* **2019**, *57*, 482–498. [CrossRef]
31. Ma, W.; Zhang, J.; Yu, J. Non-intrusive reduced order modeling for flowfield reconstruction based on residual neural network. *Acta Astronaut.* **2021**, *183*, 346–362. [CrossRef]
32. Zhu, X.; Cai, Z.; Wu, J.; Cheng, Y.; Huang, Q. Convolutional neural network based combustion mode classification for condition monitoring in the supersonic combustor. *Acta Astronaut.* **2019**, *159*, 349–357. [CrossRef]
33. Park, S.-Y.; Ahn, J. Deep neural network approach for fault detection and diagnosis during startup transient of liquid-propellant rocket engine. *Acta Astronaut.* **2020**, *177*, 714–730. [CrossRef]
34. Yonekura, K.; Hattori, H. Framework for design optimization using deep reinforcement learning. *Struct. Multidiscip. Optim.* **2019**, *60*, 1709–1713. [CrossRef]
35. Ling, J.; Templeton, J. Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty. *Phys. Fluids* **2015**, *27*, 085103. [CrossRef]
36. Zhang, Y.; Sung, W.J.; Mavris, D.N. Application of convolutional neural network to predict airfoil lift coefficient. In Proceedings of the 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, USA, 8–12 January 2018; p. 1903.
37. Yu, B.; Xie, L.; Wang, F. An improved deep convolutional neural network to predict airfoil lift coefficient. In *International Conference on Aerospace System Science and Engineering*; Springer: Singapore, 2019; pp. 275–286.
38. Hui, X.; Bai, J.; Wang, H.; Zhang, Y. Fast pressure distribution prediction of airfoils using deep learning. *Aerosp. Sci. Technol.* **2020**, *105*, 105949. [CrossRef]
39. Peng, J.-Z.; Chen, S.; Aubry, N.; Chen, Z.-H.; Wu, W.-T. Time-variant prediction of flow over an airfoil using deep neural network. *Phys. Fluids* **2020**, *32*, 123602. [CrossRef]
40. Li, K.; Kou, J.; Zhang, W. Deep neural network for unsteady aerodynamic and aeroelastic modeling across multiple Mach numbers. *Nonlinear Dyn.* **2019**, *96*, 2157–2177. [CrossRef]
41. 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. *Comput. Fluids* **2020**, *198*, 104393. [CrossRef]
42. Tompson, J.; Schlachter, K.; Sprechmann, P.; Perlin, K. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*; PMLR: Sydney, Australia, 2017; pp. 3424–3433.
43. Murata, T.; Fukami, K.; Fukagata, K. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *J. Fluid Mech.* **2020**, *882*. [CrossRef]
44. Kou, J.; Zhang, W. An approach to enhance the generalization capability of nonlinear aerodynamic reduced-order models. *Aerosp. Sci. Technol.* **2016**, *49*, 197–208. [CrossRef]
45. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

46. Keskar, N.S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; Tang, P.T.P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv* **2016**, arXiv:1609.04836.
47. Krist, S.L. *CFL3D User's Manual*; version 5.0; National Aeronautics and Space Administration: Washington, DC, USA; Langley Research Center: Hampton, VA, USA, 1998.
48. Ladson, C.L. *Effects of Independent Variation of Mach and Reynolds Numbers on the Low-Speed Aerodynamic Characteristics of the NACA 0012 Airfoil Section*; National Aeronautics and Space Administration, Scientific and Technical Information Division: Washington, DC, USA, 1988.
49. Pires, O.; Munduate, X.; Ceyhan, O.; Jacobs, M.; Madsen, J.; Schepers, J.G. Analysis of the high Reynolds number 2D tests on a wind turbine airfoil performed at two different wind tunnels. *J. Phys. Conf. Ser.* **2016**, *749*, 12014. [CrossRef]
50. Yilmaz, E.; German, B. A convolutional neural network approach to training predictors for airfoil performance. In Proceedings of the 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Denver, CO, USA, 5–9 June 2017; p. 3660.
51. Sethian, J.A. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci. USA* **1996**, *93*, 1591–1595. [CrossRef] [PubMed]
52. Qiang, L.; Zhenbing, L.; Xiong, D.; Lin, W.; Yan, Z. Numerical investigation on flow field characteristics of dual synthetic cold/hot jets using POD and DMD methods. *Chin. J. Aeronaut.* **2020**, *33*, 73–87.
53. Vitkovicova, R.; Yokoi, Y.; Hyhlik, T. Identification of structures and mechanisms in a flow field by POD analysis for input data obtained from visualization and PIV. *Exp. Fluids* **2020**, *61*, 171. [CrossRef]