Time-variant prediction of flow over an airfoil using deep neural network

Cite as: Phys. Fluids **32**, 123602 (2020); https://doi.org/10.1063/5.0022222
Submitted: 20 July 2020 . Accepted: 10 November 2020 . Published Online: 02 December 2020

Jiang-Zhou Peng (彭江舟), **Siheng Chen** (陈思衡), **Nadine Aubry**, **Zhi-Hua Chen** (陈志华), and **Wei-Tao Wu** (吴威涛)

Physics of Fluids ARTICLE scitation.org/journal/phf

Time-variant prediction of flow over an airfoil using deep neural network

Cite as: Phys. Fluids 32, 123602 (2020); doi: 10.1063/5.0022222
Submitted: 20 July 2020 • Accepted: 10 November 2020 •
Published Online: 2 December 2020

Jiang-Zhou Peng (彭江舟),[1,a)] Siheng Chen (陈思衡),[2,b)] Nadine Aubry,[3,c)] Zhi-Hua Chen (陈志华),[1,a)] and Wei-Tao Wu (吴威涛)[4,d)]

AFFILIATIONS

[1]Key Laboratory of Transient Physics, Nanjing University of Science and Technology, Nanjing 210094, China
[2]Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Floor 8, Cambridge, Massachusetts 02139, USA
[3]Department of Mechanical Engineering, Tufts University, Medford, Massachusetts 02155, USA
[4]School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

[a)]pengjz@njust.edu.cn and chenzh@mail.njust.edu.cn
[b)]schen@merl.com
[c)]nadine.aubry@tufts.edu
[d)]**Author to whom correspondence should be addressed:** weitaowwtw@njust.edu.cn

ABSTRACT

In this article, we propose an unsteady data-driven reduced order model (ROM) (surrogate model) for predicting the velocity field around an airfoil. The network model applies a convolutional neural network (CNN) as the encoder and a deconvolutional neural network (DCNN) as the decoder. The model constructs a mapping function between temporal evolution of the pressure signal on the airfoil surface and the surrounding velocity field. For improving the model performance, the input matrix is designed to further incorporate the information of the Reynolds number, the geometry of the airfoil, and the angle of attack. The DCNN works as the decoder for better reconstructing the spatial and temporal information of the features extracted by the CNN encoder. The training and testing datasets of flow fields under different conditions are obtained by solving the Navier–Stokes equations using the computational fluid dynamics method. After model training, the neural network based ROM shows accurate and dramatically fast predictions on the flow field of the testing dataset with extended angles of attack and Reynolds numbers. According to the current study, the neural network-based ROM has exhibited attractive potentials on ROM of the unsteady fluid dynamic problem, and the model can potentially serve on investigating flow control or optimization problems in the future.

I. INTRODUCTION

During the course of an aircraft's flight, there may exist various forms of disturbances that affect its aerodynamics. Especially, for a small and light aircraft, the disturbance may induce stall and necessitate some form of flow control to regulate the aircraft.[1] The traditional way of solving the fluid mechanics problem relies on physics modeling, usually in the form of differential equations. Most of these theoretical models are high-dimensional, multi-scale, and nonlinear and therefore can only be solved numerically/computationally. Numerical computation is able to produce high-accuracy results, but the computational cost is still high, especially for engineering-scale problems. The time required to obtain a numerical result is often measured in hours or even days; this can be more expensive for problems of optimization and flow control, which usually require several hundreds of simulations. One of the effective alternatives to these high-fidelity computational fluid dynamics (CFD) simulations is the reduced order model (ROM), which is very computationally efficient.[2]

Machine learning (ML) enabled data-driven reduced order modeling (ROM) has been a rising research interest in fluid mechanics.[3] The rapid expansion in data science and machine learning techniques has opened new doors in understanding unsolved problems in aerodynamics and fluid mechanics.[4] In the last few years, researchers in fluid mechanics have shown great successes in learning representations from data, especially for those quasi-steady state

problems that relatively contain less features/patterns. One of the effective uses to these high-fidelity CFD simulations is the reduced order model based on the convolutional neural network (CNN). Bhatnagar *et al.*[5] developed a model for predicting flow over airfoils, and the mean square error is less than 10% for different airfoil shapes, angles of attack (AOA), and Reynolds numbers. Using the methods of machine learning and inverse modeling, Singh *et al.*[6] reconstructed predictive models for turbulent separated flows over airfoils by effectively using limited data generated from physical experiments. Using the data generated by CFD with the Spallart–Allmaras (SA) model, Zhu *et al.*[7] reconstructed a mapping function between the turbulent eddy viscosity and the mean flow variables for subsonic flows around airfoils. Sekar *et al.*[8] developed a data-driven model for predicting incompressible laminar steady flow over airfoils with different geometries, Reynolds numbers, and angle of attack, based on a deep CNN and deep multilayer perceptron. Lui and Wolf[9] presented a numerical methodology for construction of ROM of fluid flows through the combination of flow modal decomposition and regression analysis. In those ROM methods, the low-dimensional modes of the fluid dynamics system are extracted using CNN, and the evolution of the reduced order dynamical system (mode coefficients) is regressed by the multilayer of CNN for better capturing the nonlinear features of the dynamic system.

For flight control, instead of the quasi-steady state, an unsteady model is required due to the importance of the response of the flow field to the time-varying disturbances. Once the evolution of the aerodynamic flow field is able to be predicted by the model in time, the control on the flow becomes theoretically feasible.[10] Recently, some pioneering investigations have been performed on modeling the spatial–temporal evolution of the flow fluid using convolutional layers combined with fully connected layers. Miyanawala and Jaiman[11] presented a reduced order model for the unsteady flow problems by combining the CNN and fully connected network (FCN). The model shows outstanding performance on predicting flow over different shapes of bluff bodies at a low Reynolds number. Based on the CNN, Tompson *et al.*[12] developed a data-driven model for simulating real-time 2D and 3D fluid flows, where the obtained results are realistic and show good generalization. Omata and Shirayama[13] proposed a data-driven nonlinear low-dimensional representation method for analyzing the time series data of unsteady flow fields. The authors first extract the features of the flow using the CNN encoder and then project these features using principle component analysis (PCA) to achieve the principle components of the flow, and finally, the spatio-temporal structure of the flow is visualized by plotting the temporal trajectory of the principle components. Jin *et al.*[14] proposed deep CNN layers as the encoder with FCN layers as the decoder to predict the unsteady velocity field around a circular cylinder using the pressure distribution on the cylinder. The time series of the pressure distribution on the cylinder is converted to be two-dimensional datasets as the input of the CNN. These structures perform well in the prediction of spatial–temporal evolution of the flow fluid.[15]

In this work, we propose a neural network based reduced order model (ROM) decoder to predict the unsteady flow over an airfoil, and the model constructs a mapping function between temporal evolution of the pressure signal on the airfoil and the surrounding velocity field (corresponding work[16]). Specifically, for better extracting and utilizing the intrinsic features of the flow dynamic system to reconstruct/predict the flow field, the proposed ROM applies a convolutional neural network (CNN) as the encoder to reduce the dimension and extract instinct feature of the flow system[17,18] since the CNN has a great potential for feature extraction of flow field in lower dimensions;[19] in the proposed model, a deconvolutional neural network (DCNN) is used as the decoder to reconstruct the flow fields, instead of using FCN. The latter has the disadvantage of squashing the original two-dimensional matrix into a one-dimensional vector, which may lead to the loss of the spatial information. For further improving the performance of the model, we also feed the Reynolds number and the geometry of the airfoil into the network. Different flow conditions are investigated for testing the performance of the model. The paper is organized as follows: In Sec. II, we introduce the methods, including preparation of the datasets, network architecture of the model, and a detailed description of the algorithm. Subsequently, in Sec. III, the predictions of the trained model in different flow conditions are performed, and in Sec. IV, we discuss some physical interpretation for the network model. Finally, in Sec. V, we summarize current work, and we give an outlook toward some potential directions for future exploration.

II. METHODS

In this work, we build a reduced order model (ROM) mapping the temporal evolution of surface pressure on an airfoil to the surrounding velocity field under different conditions, including the angle of attack and Reynolds number. The proposed network model first encodes the temporal series of the pressure distribution, the shape of the airfoil, and Reynolds number using a convolutional neural network (CNN) and then decodes the features back to the velocity field using a deconvolutional neural network (DCNN). A combined CNN–DCNN model excels at "learning" and "utilizing" essential features from data for prediction. The network architecture and its training are implemented using TensorFlow. The training and testing datasets are generated by numerically solving the incompressible Navier–Stokes equations.

A. Design of neural network model

1. Model workflow

The network model can be considered as a special methodology to regress an arbitrary non-linear function,[20]

$$\hat{\boldsymbol{u}} = f(\boldsymbol{X}, \boldsymbol{\theta}), \tag{1}$$

where $\hat{\boldsymbol{u}}$ is the velocity field predicted by the network model, $\boldsymbol{X}$ is the input matrix, and $\boldsymbol{\theta}$ is the learnable model parameter of the network. For a given input $\boldsymbol{X}$, we want to approximate the output $\hat{\boldsymbol{u}}$ using the function (f) as closely as possible to the velocity field ($\boldsymbol{u}$) of CFD simulation. The function (f) here is exactly the network model.

Figure 1 depicts the overall graph of the proposed network, which shows the learning strategy of the proposed neural network model for predicting the time series of the velocity field from the surface pressure data. The second block of the framework, shuffling, is applied to disrupt the input data. A disordered dataset ensures that the neural network can randomly select the direction of the

FIG. 1. The total graph of the deep learning prediction model configuration.

gradient optimization; thus, the local optimization is avoided efficiently. In the third block, the structure of the original dataset is preprocessed to be a proper form. The structure of the training datasets is important for guiding the network to learn the intrinsic features embedded in the data.[1] Therefore, we standardize the dataset with zero mean and unit standard deviation, and after the standardization, the training data are further normalized into a range from −1 to 1 for improving the stability and the speed of the training process. Then, the preprocessed data are sent into the core block: the CNN–DCNN network. The purpose of the CNN–DCNN block is to search and magnify the intrinsic features from the dataset, which can be used to discern spatial–temporal series of the flow system.[21] Eventually, the output of the CNN–DCNN block is sent to a reshape block; the operation here stacks only the output of previous layer to form a new vector so that the flow field can be predicted in an orderly manner.

2. Design of neural network

The proposed network architecture uses CNN to extract spatial and temporal information from the input data and then applies the DCNN to reconstruct the entire velocity field. Figure 2 shows the structure and components of the CNN–DCNN model. The CNN–DCNN model is composed of four layers of encoders and four layers of decoders. The deconvolutional (decoding) part mirrors the architecture of the convolutional layer. Such an inverse operation (deconvolution) unravels the high-level features encoded by the CNN. Furthermore, as Table I indicates, every convolutional layer is composed of $2^{(3+l)}$ convolutional filters, where l corresponds to the index of the CNN layer.

Each CNN layer involves the convolution and nonlinear activation operations and can be expressed mathematically as

$$\boldsymbol{a}_l = \sigma(\boldsymbol{W}_l * \boldsymbol{a}_{l-1} + \boldsymbol{b}_l), \tag{2}$$

where $\boldsymbol{a}_l$ is the output of the lth layer, σ denotes the nonlinear activation function, $*$ is the convolutional operator, $\boldsymbol{b}_l$ is the bias, and

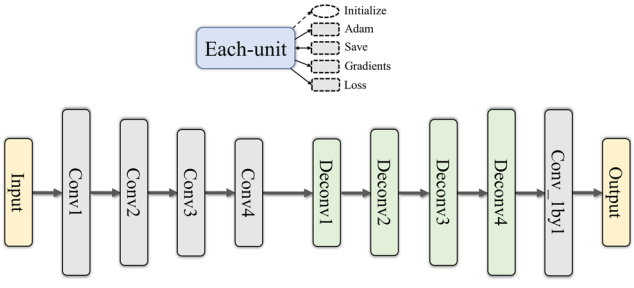

FIG. 2. Architecture of the CNN–DCNN model. "Conv" denotes the convolutional layer, "Deconv" denotes the deconvolution layer, "Conv_1by1" represents the one by one convolutional operation, and "Each-unit" shows the operations in each unit during model training.

TABLE I. Parameters of the convolutional layer encoder, where $w \times h$ denotes the size of convolution kernel ($\boldsymbol{F}$), n_F denotes the number of $\boldsymbol{F}$, and $N_1 \times N_2 \times \boldsymbol{d}$ is the size of the output after convolution.

Name of layer	Executing operation $w \times h \times n_F$/stride	Shape $N_1 \times N_2 \times d$
Input of model	...	87 × 87 × 3
Conv1	5 × 5 × 16/1	83 × 83 × 16
Conv2	3 × 3 × 32/2	41 × 41 × 32
Conv3	3 × 3 × 64/2	20 × 20 × 64
Conv4	3 × 3 × 128/2	9 × 9 × 128
Deonv1	3 × 3 × 64/2	20 × 20 × 64
Deconv2	3 × 3 × 32/2	41 × 41 × 32
Deconv3	3 × 3 × 16/2	83 × 83 × 16
Deconv4	5 × 5 × 3/1	87 × 87 × 3
Conv_1by1	1 × 1 × 1/1	87 × 87 × 1
Output	Reshape	7569 × 1

$\boldsymbol{W}_l$ is the weights or convolutional kernel. The introduction of σ is crucial for the neural network to possess non-linearity. For the activation function, in this paper, we apply the exponential linear unit (ELU) activation function,

$$\sigma(x) = \begin{cases} c(e^x - 1), & x < 0 \\ x, & x \geq 0, \end{cases} \tag{3}$$

where c is an adjustable constant. The advantage of ELU is that the negative input could also be activated by ELU, thus preventing deactivation of neural nodes. Note that because the final prediction of the network needs to be a continuous regression, the last layer of the entire network has no activation function.

The convolution operation is the key to the whole network model. The schematic of the convolution operation is shown in Fig. 3, where the 4 × 4 light purple matrix is the convolutional kernel and the gray matrix is the input matrix. Besides the size of the kernel, F, the convolutional output is also influenced by kernel stride, S, and the padding size, P. The padding operation adds zeros around the border of the input matrix, and the kernel stride controls the sliding step size of the kernel. The size of the output after convolutional operation can be calculated as[22]

$$H_{l+1} = \frac{H_l - F + 2P}{S} + 1, \tag{4}$$

where H_l is the size of feature map at lth layer. In this paper, zero padding layer is used in this article ($P = 0$), and the first convolutional kernel is equipped with $F_1 = 5$, $S_1 = 1$, while others are designed as $F = 3$, $S = 2$. From the above equation, it can be seen that after several convolutional operations, the size of the original input can be reduced significantly; thus, it becomes easy for the network to learn.

B. Model training

The model training is an iterative process, which continuously minimizes the error/loss between the predicted ($\hat{\boldsymbol{u}}$) and the target ($\boldsymbol{u}$) outputs,[23,24] for obtaining the optimal model parameters, $\boldsymbol{\theta}$. The

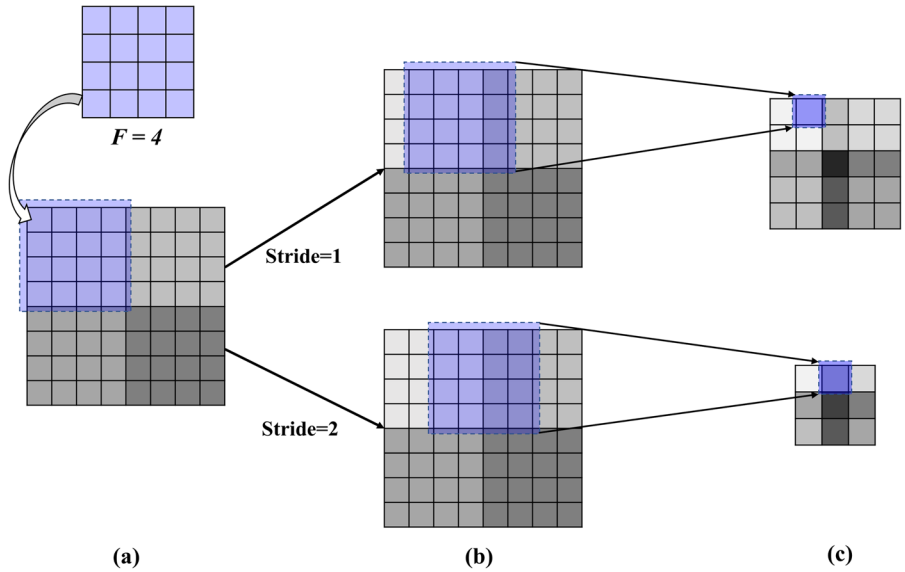

FIG. 3. Schematic of the convolution operation. The light purple matrix denotes a 4 × 4 convolutional kernel. (a) represents the beginning of the convolution, (b) represents the different stride of convolution operation, and (c) shows the result of convolution operation.

loss/error function is defined as

$$J = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{u}_n - \hat{\boldsymbol{u}}_n)^2 + \lambda||\boldsymbol{W}||_2, \tag{5}$$

where $\boldsymbol{u}_n$ indicates the result by CFD simulation, $\hat{\boldsymbol{u}}_n$ is the result predicted by the network model, $\boldsymbol{W}$ denotes all the weight of the network layers, N is the number of training data, λ is the regularization coefficient, and $\lambda||W||_2$ is the L2 regularization for preventing model overfitting. The neural network is trained by minimizing the above loss function over the training dataset. In order to improve the computational efficiency and the quality of the model, a mini-batch based learning strategy is used during the training.[25]

The backpropagation method with the Adaptive Moment Estimation (Adam) method is implemented as the training algorithm. Adam is a stochastic gradient descent variant that computes the adaptive learning rate for each parameter. The main idea of the scheme is dependent upon the establishment of the first (m_t) and second (v_t) order moments of the gradients of the loss function,[26,27]

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t, \tag{6}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2, \tag{7}$$

where β_1 and β_2 are the exponential decay rates for the moment estimates, g_t denotes the gradients of the loss function with respect to the parameters of the network, and m_t and v_t are the estimates of the first moment (the mean) and the second moment (the variance), respectively. These two moments control the update direction of the model weight and the step length (learning rate), respectively. Since m_t and v_t are initialized using zero vectors, from above equations, it can be seen that the Adam optimizer is biased toward zero, especially during the initial time steps. Therefore, the following correction is proposed to counteract these biases (note that the superscript t of β is represented exponentially):

$$m_t^{corrected} = \frac{m_t}{1-\beta_1^t}, \tag{8}$$

$$v_t^{corrected} = \frac{v_t}{1-\beta_2^t}. \tag{9}$$

Finally, the model parameters vector, $\boldsymbol{\theta}$, is iteratively updated through

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha\frac{m_t^{corrected}}{\sqrt{v_t^{corrected}} + \varepsilon}, \tag{10}$$

where α is the learning rate and ϵ is a number to avoid singularity. The workflow of the model training is summarized in the following:

1. Select the proper preprocessed CFD results as the training dataset.
2. From the training dataset, split out the validation dataset (10% of the training dataset).
3. Prepare the test set, which is different from the training set and will not be seen by the network model during the training.
4. Train the model using back propagation with the Adam algorithm, and the iteration number is limited to be 20 000.
5. Evaluate the performance of the trained neural network model using the test dataset by comparing the predicted results of the network model with CFD simulation.

C. Preparation of dataset

A good dataset is an important aspect for deep learning to achieve better training and prediction performance. Lack of enough training data, which has nothing to do with the quality of the network structure, can endanger the accuracy of the network model.[28] Therefore, a high-quality training dataset should be ensured for any

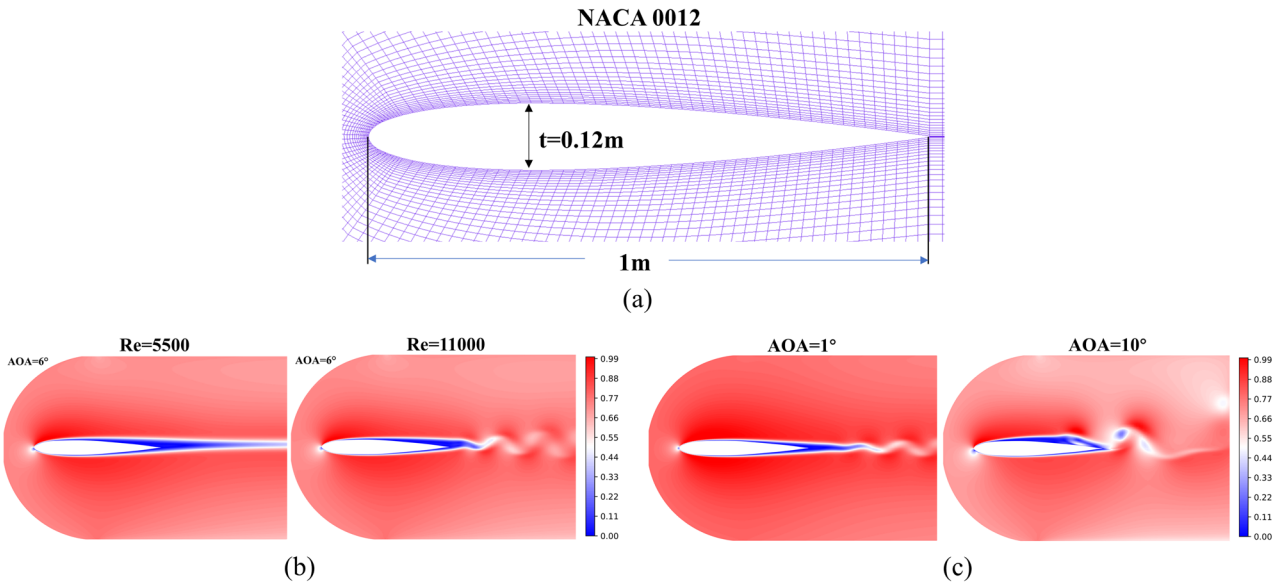

FIG. 4. Instantaneous flow fields over the airfoil (a) under different Reynolds numbers and angles of attack by CFD simulation. (b) Flow fields simulation results at AOA = 6° and (c) flow fields simulation results at Re = 11 000.

network modeling as the first priority. The pressure distribution on the airfoil and the flow field around the airfoil are obtained by numerical solving the Navier–Stokes equations, and the shape of the airfoil that also includes the information of the angle of attack is represented by a binary image. The present approach contains two steps of data preparation: the flow simulation by CFD and the parameterization of the flow condition (see Subsections II C 1 and II C 2 for details).

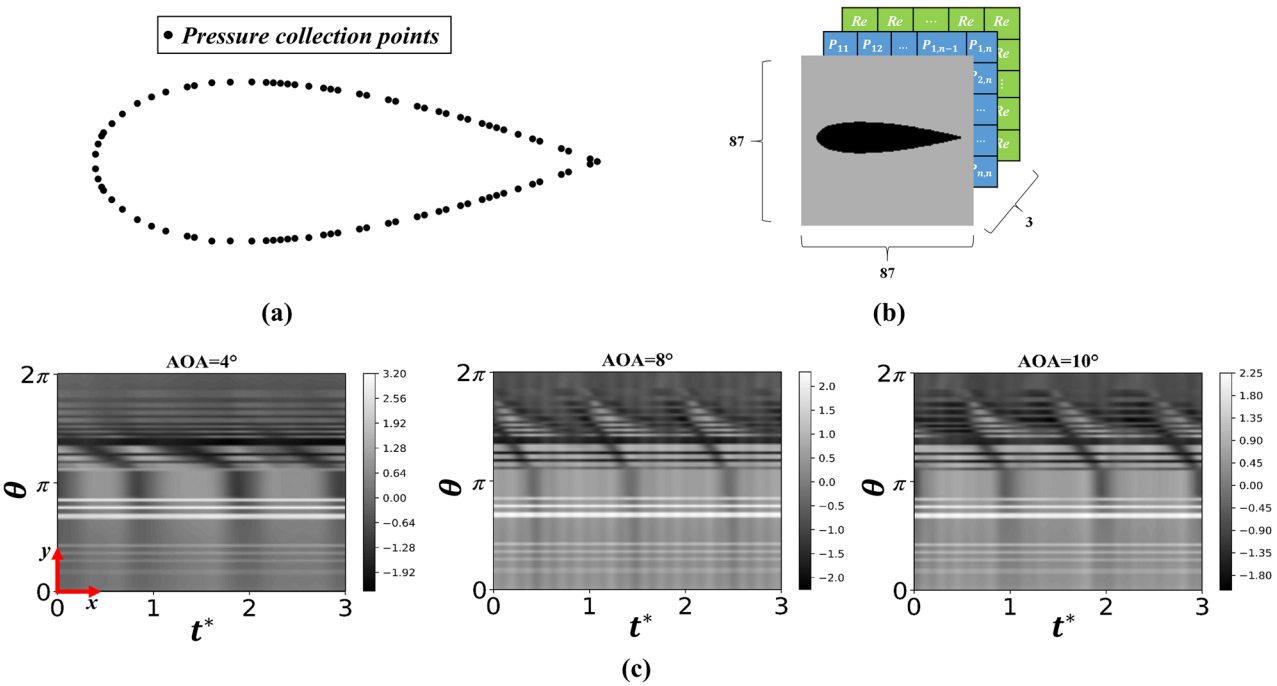

FIG. 5. (a) Schematic of the distribution of the sampled points on the airfoil surface. (b) Schematic of the structure of the input matrix. (c) Distribution of the pressure signal in the spatial–temporal domain.

1. Governing equations and CFD simulation

We study the incompressible, isothermal linear/Newtonian fluid flow, and the governing equations are

$$\nabla \cdot \boldsymbol{u} = 0, \tag{11}$$

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) = \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b}, \tag{12}$$

where

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \mu\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right), \tag{13}$$

where $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t)$ is the velocity field, $\boldsymbol{b}$ is the body force, ρ is the density, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, p is the static/spherical pressure, and μ is the dynamic viscosity of the fluid. The dimensionless number Reynolds number is defined as $Re = \rho U_\infty L/\mu$, where U_∞ is the freestream velocity and L is the chord length of airfoil. The above equations are numerically solved using OpenFOAM. Specifically, we consider a two-dimensional laminar flow over an airfoil ($Re < 1 \times 10^5$[29]). The flow domain over the NACA0012 airfoil is discretized using a C grid of 116-by-116 points. Figure 4 displays several typical instantaneous flow fields over the airfoil by CFD simulation. The training dataset contains the cases of Re = [6000, 7000, 8000, 9000, 10 000], and for each Reynolds number, simulations are also performed with several different angles of attack (AOA) = [2°, 3°, 4°, 5°, 6°, 7°, 8°, 9°]; the testing dataset contains the cases of Re = [5500, 8500, 11 000] and AOA of [0°, 1°, 8.5°, 10°].

In order to capture the enough temporal information from the data, each set of pressure signals on the airfoil surface is sampled with a time interval, δt. n of these pressure signals are combined as a

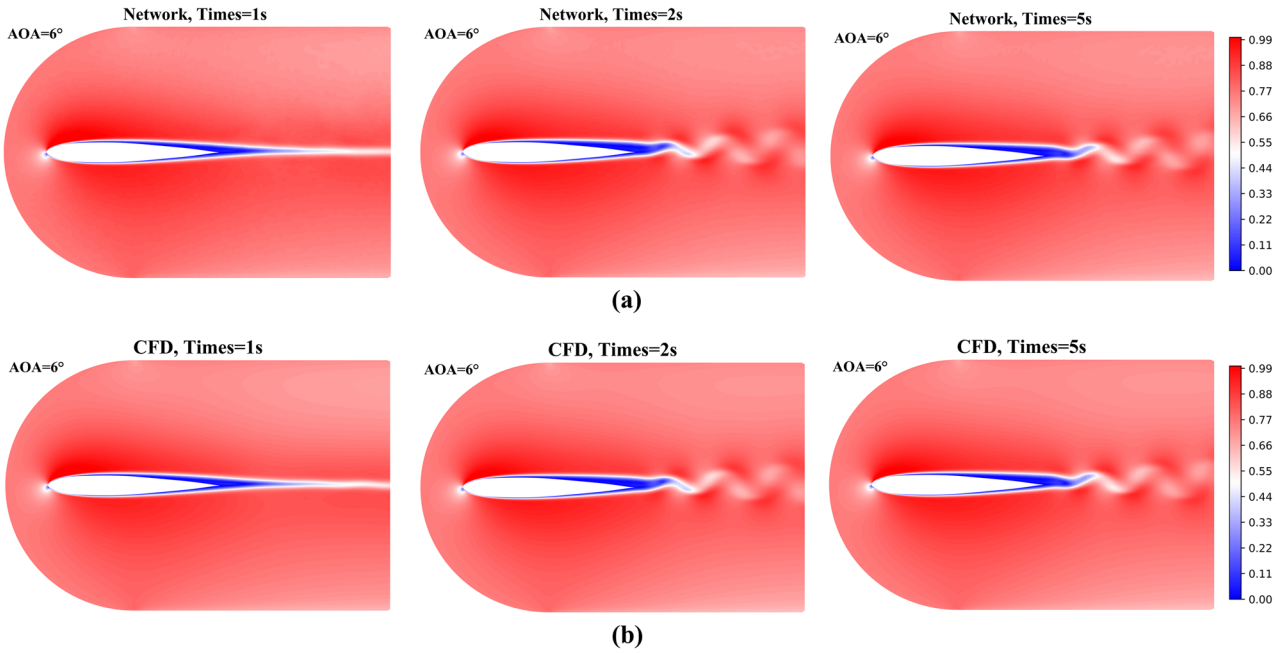

FIG. 6. Evolution of the flow field predicted by the CNN–DCNN model with angle of attack of 6° and Reynolds number of 10 000. (a) Evolution of the flow field predicted by the CNN–DCNN model and (b) evolution of the flow field simulated by OpenFOAM.

TABLE II. Time consumption of the flow prediction by the CNN–DCNN model and CFD simulation. The time of prediction is dimensionless time $t^* = 4$. The CPU is i5-7300, and the GPU is RTX 2080ti.

		CNNs (s)			
Re number	AOA (deg)	GPU	CPU	OpenFOAM (s)	Speedup (GPU/CPU)
5500		0.546 545	2.752 32	840	1537/305
8500	6	0.545 512	2.768 289	898	1646/324
11 000		0.547 532	2.792 353	971	1773/347
	1	0.548 505	2.749 329	821	1494/298
10 000	8.5	0.549 53	2.748 62	872	1587/317
	10	0.548 533	2.762 358	893	1628/323

matrix to feed the network, and the velocity field ($\boldsymbol{u}$) is extracted per every $n\delta t$ as the truth data for training and testing; in other words, the time interval between two consecutive snapshots is $\Delta t = n\delta t$. For a more accurate temporal variant prediction of flow, Δt is designed to be less than one third of the minimum vortex shedding period of the studied cases. Hence, the integer multiples of such time interval (Δt) can be used to represent the vortex shedding period (T) under various Reynolds numbers. Furthermore, we use the vortex shedding

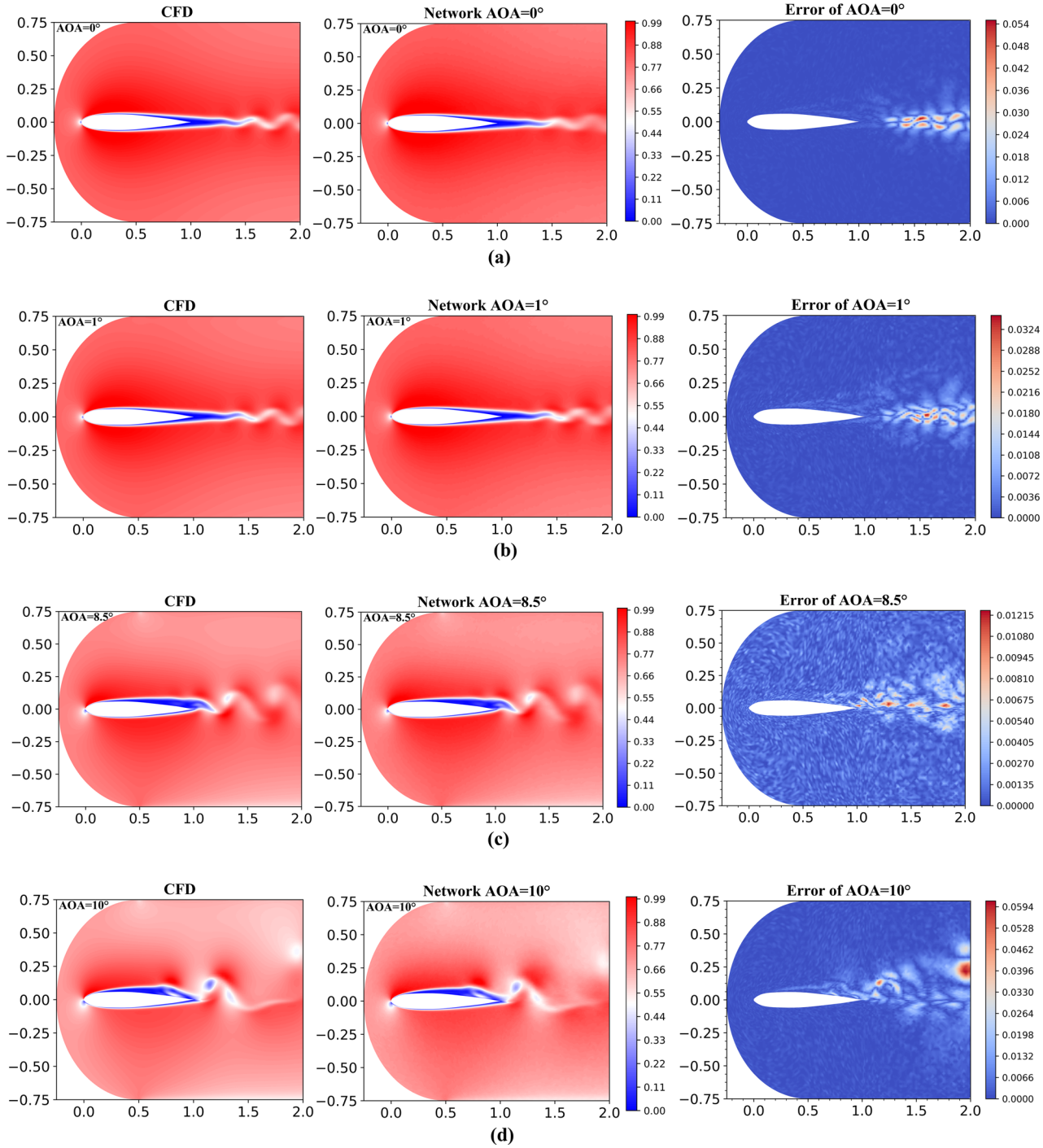

FIG. 7. Comparison of the instantaneous velocity fields by the CNN–DCNN model and CFD when Re = 10 000. (Left) CFD result, (middle) network model prediction, and (right) relative effort for (a) AOA = 0°, (b) AOA = 1°, (c) AOA = 8.5°, and (d) AOA = 10°.

period (T) as the reference time to define the dimensionless time, $t^* = t/T$, and the freestream velocity (U_∞) as the reference velocity to define the dimensionless velocity, $u^* = u/U_\infty$.

For each case with a specific Reynolds number and angle of attack, we collect 1120 training samples (the dimensionless time t^* is from 2 to 5), where one sample represents one transient flow field at a specific time, and the flow field data are organized as $\{\boldsymbol{X}^k, \boldsymbol{u}^k\}$, where k represents the kth training sample, $\boldsymbol{X}$ is the input matrix that will be discussed in Sec. II C 2, and $\boldsymbol{u}^k$ is the velocity field as the output of the network. For the training datasets, which consist of five Reynolds number and eight different angles, a total of 44 800 samples are generated. Among them, 90% of samples are used for training, and other 10% are used for validation. A batch of datasets that are outside the training data and validation data are used for testing the trained network model.

2. Parameterization of flow conditions

According to Eq. (1), the network model builds a mapping function, $\hat{\boldsymbol{u}} = f(\boldsymbol{X}, \boldsymbol{\theta})$, between the input matrix and the velocity field. The input matrix is mainly composed of the pressure distribution on the airfoil; and for further improving the performance of the model, we also include the information of the Reynolds number and the geometry. These features form three two-dimensional matrices and are stacked to each other to construct the input matrix,

$$\boldsymbol{X} = [\boldsymbol{C_p}, \boldsymbol{Re}, \boldsymbol{G}], \tag{14}$$

where $\boldsymbol{Re}$ is the dimensionless Reynolds number, $\boldsymbol{G}$ is a two-dimensional binary image (matrix) where the airfoil is represented by 1 and the flow field region is represented by 0, and $\boldsymbol{C_p}$ is the wall pressure sampled from the surface of the airfoil [see Fig. 5(a) for the distribution of the sampled points]. In each binary image (matrix), $\boldsymbol{G}$, the airfoil shape is tilted at different angles, which corresponds to different angles of attack. For capturing the temporal information, $\boldsymbol{C_p}$ is organized as follows:

$$\boldsymbol{C_p} = \left[\boldsymbol{C_p^1}, \boldsymbol{C_p^2} \cdots \boldsymbol{C_p^i} \cdots \boldsymbol{C_p^N}\right], \tag{15}$$

where $\boldsymbol{C_p^i}$ ($i = 1, 2, \ldots, N$) is the wall pressure coefficient at the ith snapshot. Figure 5(a) shows the schematic of the distribution of the points where the pressure signal sampled from at each snapshot. The size of the matrix, $\boldsymbol{C_p}$, is $N_1 \times N_2$, where N_1 is equal to the number of the sampled points of the pressure signal on the airfoil and N_2 is

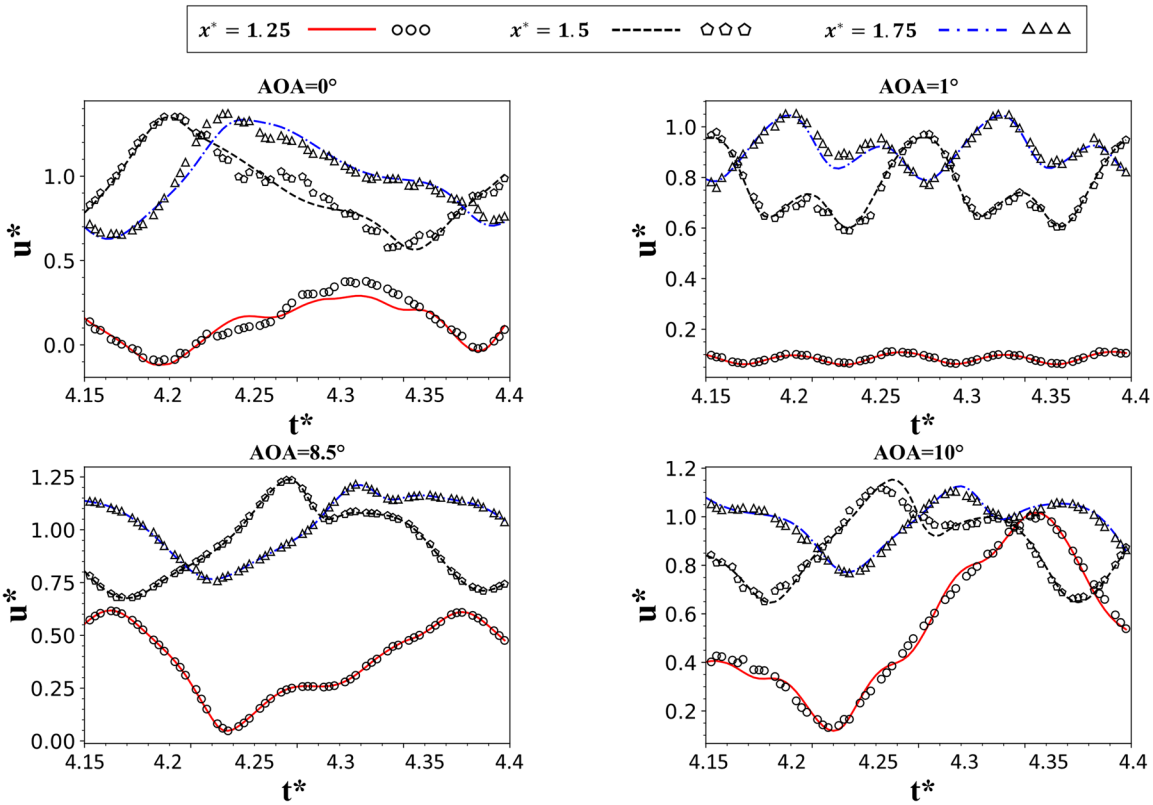

FIG. 8. Comparisons of velocity evolution in the wake region between the CNN–DCNN model predictions and CFD results ($t^* = \frac{t}{T}$). In the figures, symbols represent the results by the prediction model: x* = 1.25 (circle), x* = 1.5 (pentagram), and x* = 1.75 (triangle); lines represent the results by CFD: x* = 1.25 (red solid line), x* = 1.5 (yellow dotted-dashed line), and x* = 1.75 (blue dashed line). Both of their y coordinates are 0.2.

the number of the snapshot (interval time δt). Figure 5(b) shows the schematic of the structure of the input matrix. Figure 5(c) shows a typical distribution of the surface pressure signal, $\boldsymbol{C_p}$, which is plotted in the spatial–temporal domain. In Fig. 5(c), the x coordinate is the dimensionless time, and the y coordinate is the pressure value sampled from the surface of the airfoil. Therefore, the size of the input matrix, $\boldsymbol{X}$, is $N_1 \times N_2 \times d$ and specifically is $87 \times 87 \times 3$ in this article.

III. RESULTS

With the increasing number of training epochs, the training accuracy approaches exceeds 98.9%, and the validation accuracy approaches 97.3%. See Fig. 6 for details. The time cost of the model training using graphics processing unit (GPU) (RTX 2080ti) is about 40 min. The result preliminary illustrates the feasibility of the network, that is, the CNN–DCNN model is able to predict the two-dimensional laminar flow over airfoil accurately as an alternative to CFD method; we also compare the computational time cost of the network model and the CFD (see Table II). Using the network model, the speedup is about 300 times by central processing unit (CPU) (i5-7300) and is about 1500 by GPU (RTX 2080ti). In Secs. III A–III C, we study the capability of the designed network architecture predicting the time-varying velocity field around airfoil using the test dataset, which was never been seen by the network. We quantitatively assess the error between the network model and the ground truth that demonstrates the usability, accuracy, and effectiveness of the network.

A. Validation of the network model

To verify the generalization and extensionality of the network model, we further study the cases the model never seen, namely, the testing dataset. In detail, the testing data set includes AOA = [0°, 1°, 8.5°, and 10°], where AOA = 0°, 1°, and 10° are the extrapolation data and AOA = 8.5° is the interpolated data. The above four testing cases are studied with a *Re* of 10 000. All of the cases are laminar flow and accompanied by vortex shedding.

The predicted instantaneous flow fields of the four test cases are shown in Fig. 7. All the flow fields shown in the figures are from the same moment; therefore, the phenomena related to the effect of the angle of attack can be revealed more clearly. From the

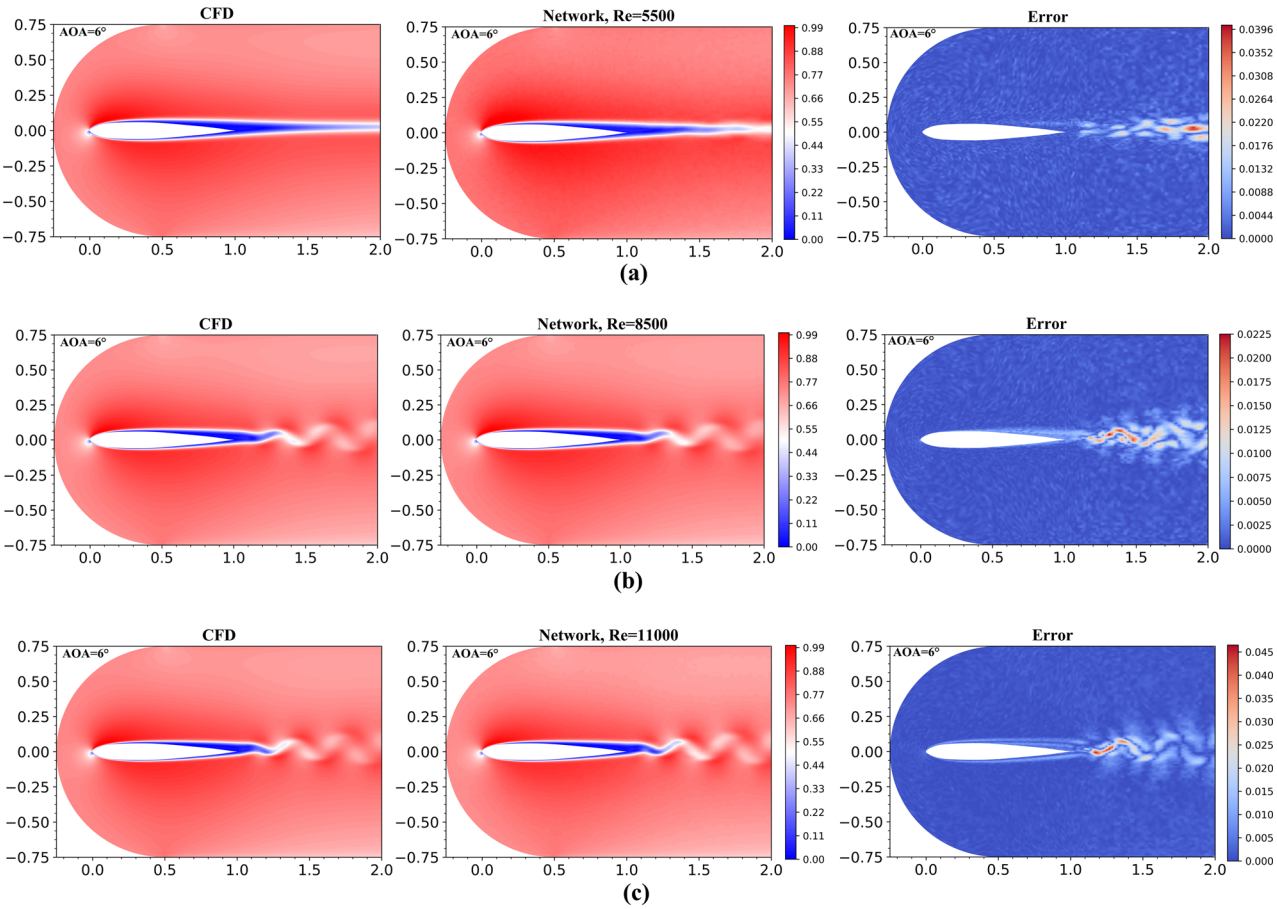

FIG. 9. Comparison of the instantaneous velocity fields by the CNN–DCNN model and CFD when AOA = 6°. (Left) CFD result, (middle) network model prediction, and (right) relative effort for (a) Re = 5500, (b) Re = 8500, and (c) Re = 11 000.

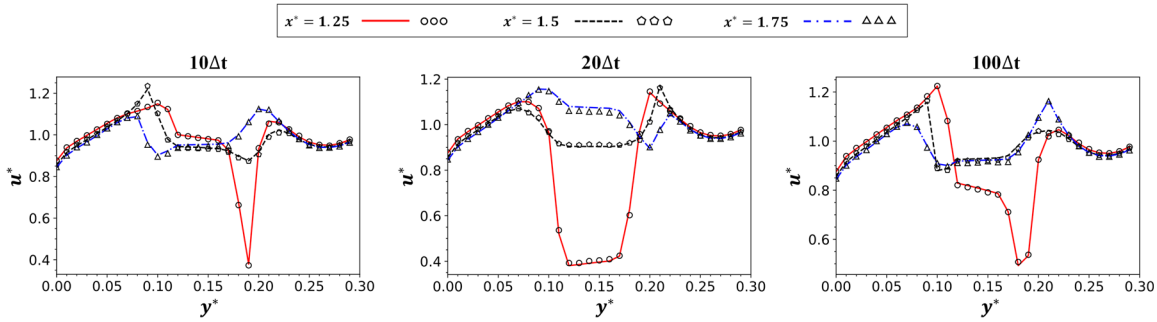

FIG. 10. Velocity profile by the network model and CFD prediction at three downstream locations, x* = 1.25 (red solid line), x* = 1.5 (black dotted-dashed line), and x* = 1.75 (blue dashed line). The three locations are indicated in Fig. 11. Circle, pentagram, and triangle represent the data of the ground truth.

results, it can be observed that the velocity field predicted by the network model shows good agreement with the CFD results. The right column of Fig. 7 shows the distributions of the error between the network model and the CFD; we can see that the maximum error locates in the wake region because the flow field there is accompanied by a large velocity gradient, namely, complex patterns. Usually, in the deep learning prediction, a large gradient is considered as one of the main causes for large error. The amplitude of the maximum error for the case of AOA = 0° is found to be the largest, while the amplitude of the maximum error's distributions of errors AOA = 8.5° is the smallest. Performance of the machine learning methods is significantly influenced by training data, and its prediction accuracy depends on the degree of the deviation of the prediction object from the training set. The training datasets include several different angles of attack (AOA) = [2°, 3°, 4°, 5°, 6°, 7°, 8°, 9°], and the case of 0° deviates greatest from the training set; therefore, the performance at 0 angle of attack is relatively poorer. Meanwhile, Fig. 8 quantitatively compared the flow field, which further verifies our discovery. Interestingly, the error for the case of AOA = 10° is larger than those of AOA = 1°, and note that both of them are extended by 1° of AOA from the training datasets. It is well known that the flow field around the airfoil is more complex as the AOA increases; that is, the above results confirm that the network model gives relatively larger error for predicting flow field with higher complexity.

On the other hand, we also further verify the generalization and extensionality of the network model with extended Reynolds numbers. The studied cases include Re = [5500, 8500, 11 000], where Re = 8500 is the interpolated case, while the other two are the extrapolation cases. All the above cases are studied with AOA = 6°. From the comparison of the prediction errors shown in Fig. 9. The maximum deviation happens when Re = 11 000. Among the studied Reynolds numbers, the case with an Re of 11 000 contains the most complex flow pattern, which confirms the perspective about the relationship between the flow complexity and prediction accuracy again.

B. Temporal characteristics of the network model

In this section, we study the ability of the CNN–DCNN model on time-varying prediction, namely, the temporal evolution of flow field. In detail, the case with an AOA of 6° and Re of 10 000 is studied at three extra points in time; all the three moments were not been seen during the model training. Furthermore, the time intervals (Δt), which are related to the sampling frequency of the pressure signal, are also different for these three cases: they are 10 times, 20 times, and 100 times larger than the time interval (Δt) used during the training, respectively. A smaller time interval implies a higher sampling frequency, thus richer input information for the network model. Figure 10 shows the comparison of temporal evolution of the velocity profile ($u^* = \frac{u}{U_\infty}$) at three different locations between the network model prediction and the CFD result. The three locations are all chosen at the downstream wake region [see Fig. 11 (each line captures 30 points)] because usually the wake region provides information of the large-scale of fluid motions. From Fig. 10, we can find that for all three locations, with the time interval of 10Δt, the velocity profile predicted by the CNN–DCNN agrees almost perfectly with the ground truth. As the time interval increases to be 20Δt, small deviations are observed in the downstream wake region $x^* = 1.75$, while $x^* = 1.25$ and $x^* = 1.5$ still perform well. The performance is not too bad even when the time interval is increased to be 100Δt.

Figure 12 shows the temporal evolution of the velocity predicted by the CNN–DCNN model and CFD at three spatial points in

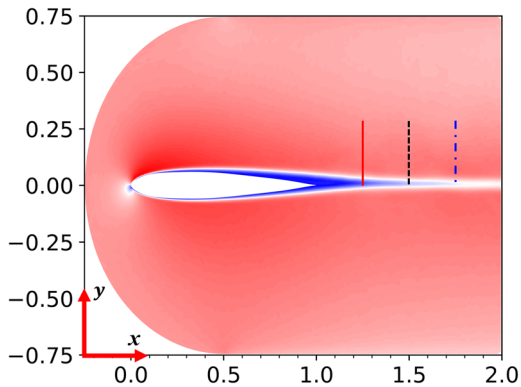

FIG. 11. Exhibition of the three studied locations in the downstream wake region.

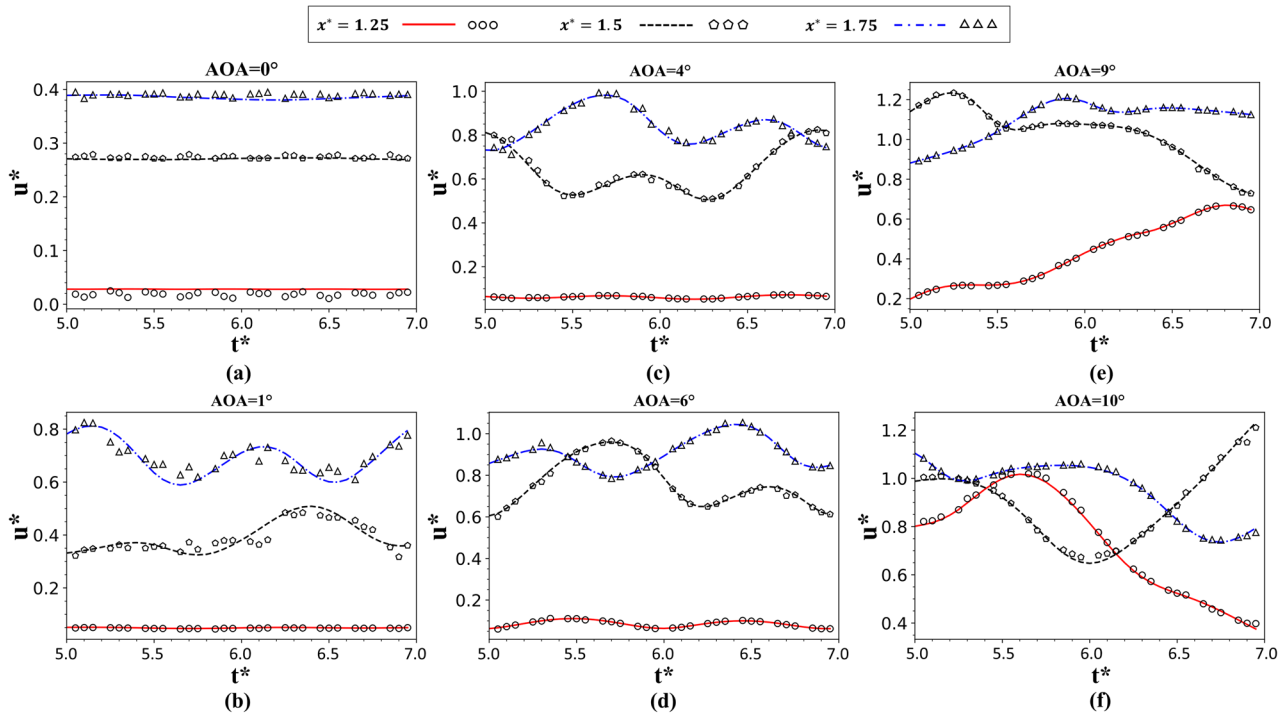

FIG. 12. Temporal evolution of the velocity predicted by the CNN–DCNN model and CFD at three spatial points in the wake region. The x coordinates of the three sampled points are the same to the three lines shown in Fig. 10, and their y coordinates are 0.2. In the figures, symbols represent the results by the network model: x* = 1.25 (circle), x* = 1.5 (pentagram), and x* = 1.75 (triangle); lines represent the results by CFD: x* = 1.25 (red solid line), x* = 1.5 (yellow dotted-dashed line), and x* = 1.75 (blue dashed line). (a) AOA = 0°, (b) AOA = 1°, (c) AOA = 4°, (d) AOA = 6°, (e) AOA = 9°, and (f) AOA = 10°.

the wake region, where the x coordinates of the three sampled points are the same to the three lines shown in Fig. 11 and their y coordinates are 0.2. Six different AOAs are studied: AOA = [0°, 1°, 4°, 6°, 9°, and 10°]. The dimensionless testing time is out of the trained time for about two vortex shedding periods. Here, the dimensionless training time is from 2 to 5, where the dimensionless time is defined as the ratio of the dimensional time to the vortex shedding period. Note that the flow field at dimensionless time from 0 to 2 has not been included into the training dataset as current study focuses on the quasi-steady state of the flow.

Obviously, the network performs well on predicting velocity fields in the testing time period, especially at an angle of attack that has been utilized during the model training, shown in (c) and (d). The additional dataset as input to the model (0°, 1°, 10°) still has

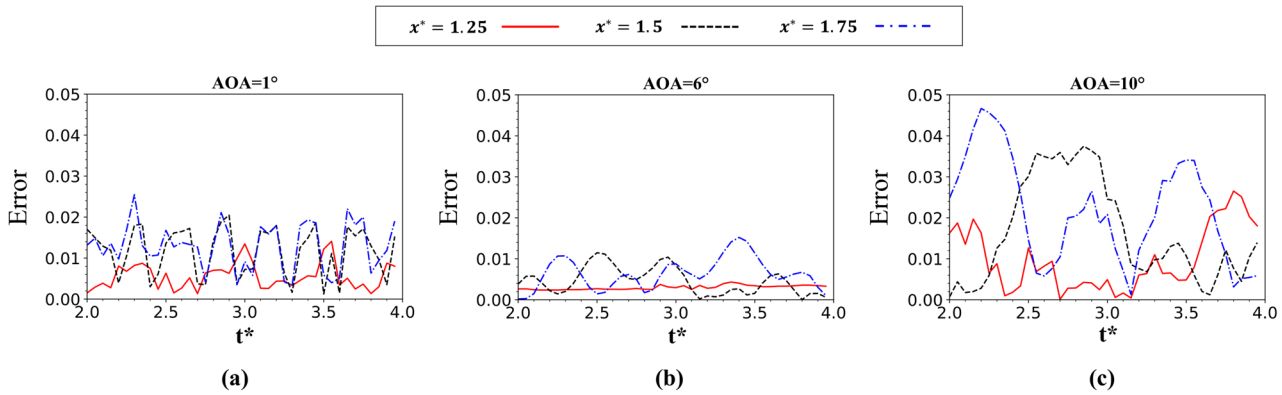

FIG. 13. The fluctuating margin comparison of three different position at wake flow behind the airfoil: x* = 1.25 (red solid line); x* = 1.5 (black dotted-dashed line); and x* = 1.75 (blue dashed line). (a) AOA = 1°, (b) AOA = 6°, and (c) AOA = 10°.

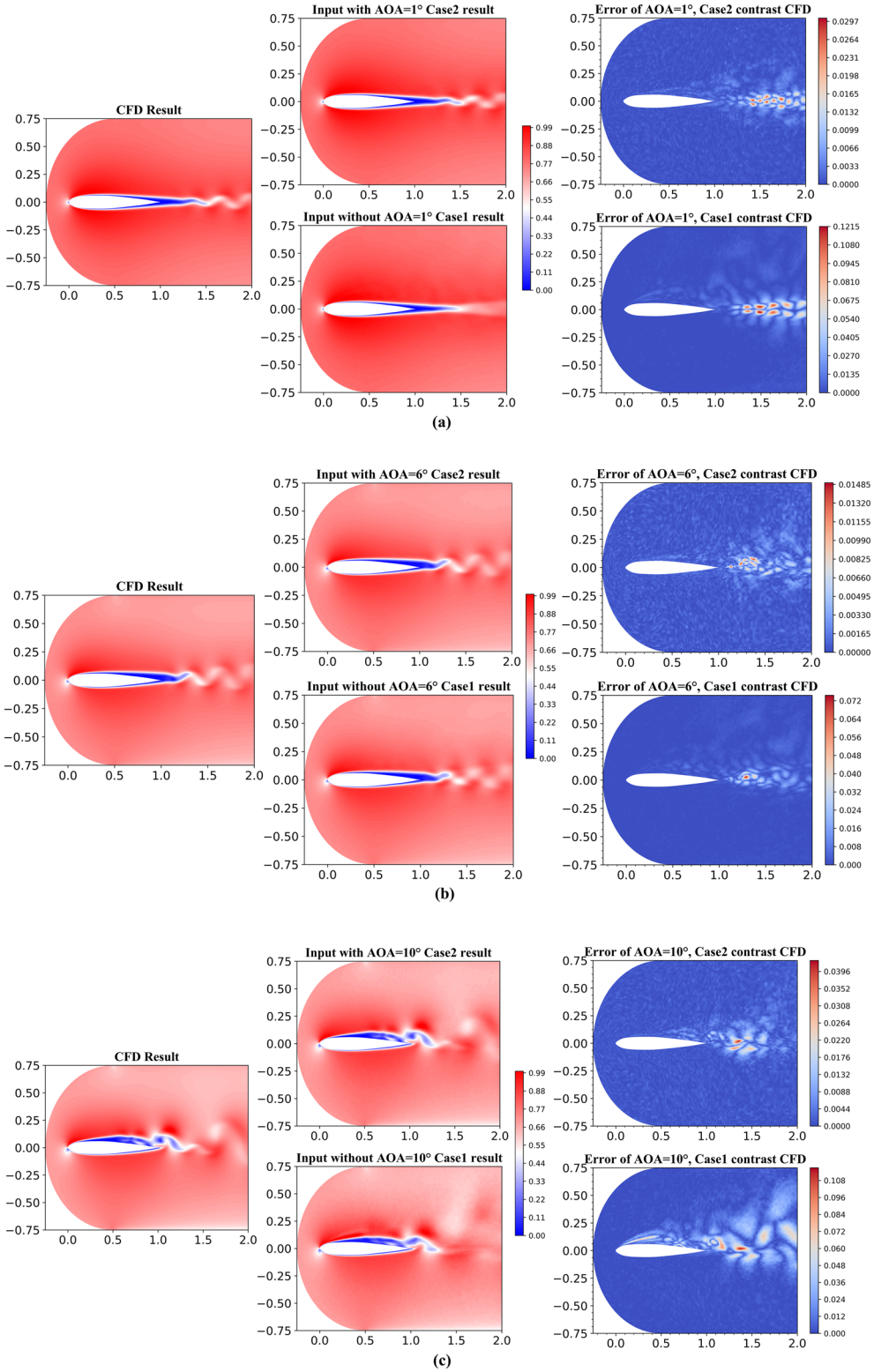

FIG. 14. Influence of the existence of the AOA in the input matrix on the performance of the network model. The velocity field by CFD (left column) and the network model (middle column) and the corresponding error distribution (right column). (a) AOA = 1°, (b) AOA = 6°, and (c) AOA = 10°.

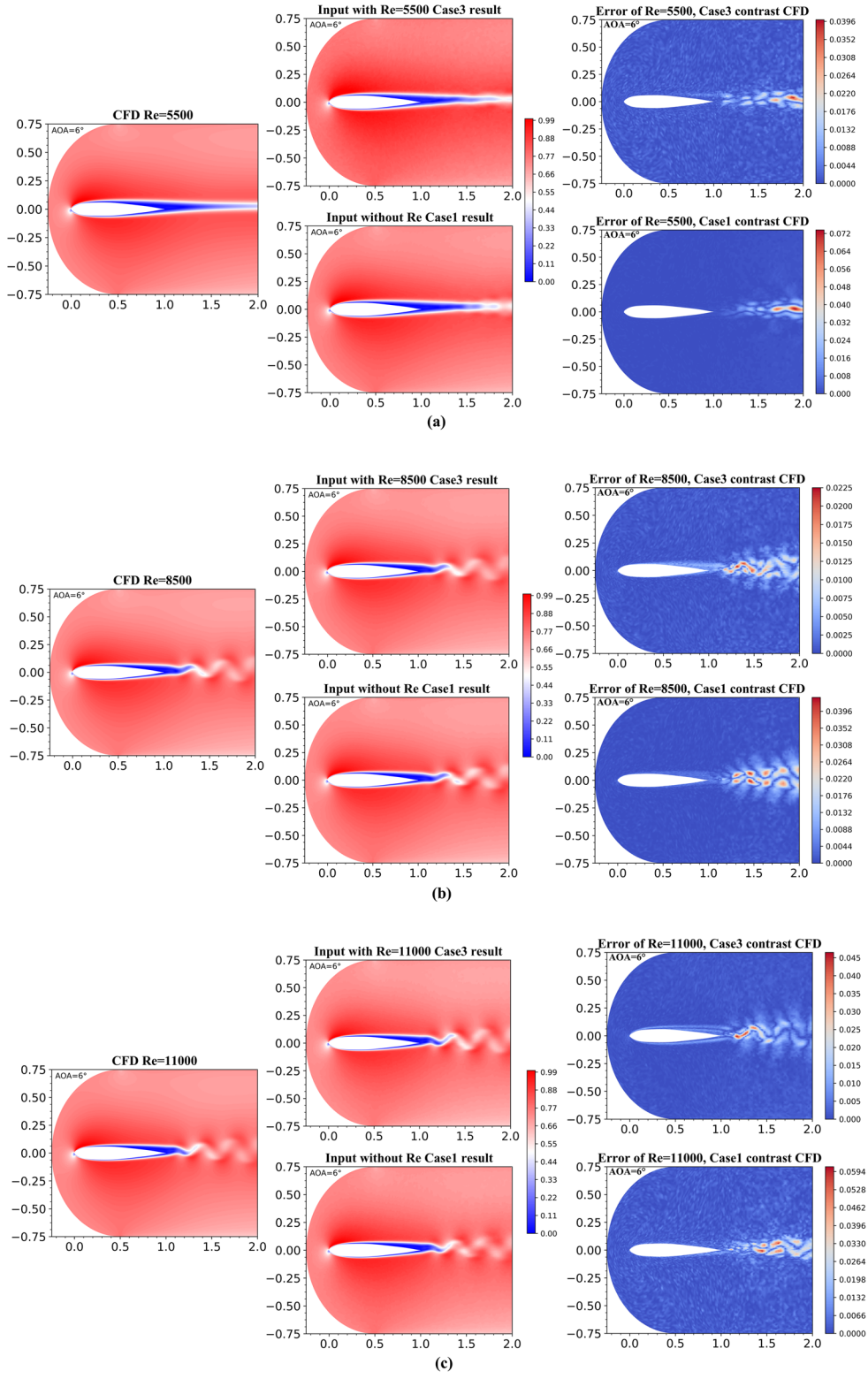

FIG. 15. Influence of the existence of the Reynolds number in the input matrix on the performance of the network model. The velocity field by CFD (left column) and the network model (middle column) and the corresponding error distribution (right column). (a) Re = 5500, (b) Re = 8500, and (c) Re = 11 000.

a larger prediction bias compared to the trained datasets as input. Especially for AOA = 0°, the flow field predicted by the network model no longer fluctuates [see the red solid line in Fig. 12(a)]. The possible solution is to increase the resolution of the sampling time. Even so, the overall prediction error is still small (Fig. 13). Therefore, we can consider that the CNN–DCNN model acquires the ability of time-varying prediction application.

In order to quantitatively visualize the performance of the network model, Fig. 13 shows the error fluctuating at the studied three positions. From the figure, we found that the maximum fluctuating occurs behind the airfoil where the flow structure changes more frequently than other two locations. This can be related to the size of the training time internal (Δt) or the kernel size and stride of the CNN filter. The case with an AOA of 6° shows obvious lower prediction error than the cases with an AOA of 1° and 10° as AOA = 6° is included in the training dataset.

C. Investigation of the influence of number of input feature classes

In this section, we study the effect of the input of the network on the model performance, and we have prepared three different cases. Case 1 informed only by the time-varying pressure signal; case 2 excludes the angle of attack; case 3 excludes the Reynolds number. Remember the complete input matrix contains the time-varying pressure signal, Reynolds number, and the binary image representing the shape of the airfoil and the angle of attack. To make the study more consistent, all the flow fields are taken from the same snapshot in time. Figure 14 shows the velocity prediction by cases 1 and 2 and the corresponding error distribution. Both of them are trained with an Re of 10 000 and the AOA of [1°, 6°, 10°]. It can be observed that the model performance is improved when introducing the AOA feature to the network. Clearly, without the instructions of physical information (AOA here), the CNN–DCNN fails in accurately predicting the flow structures in the wake region when AOA = 1° and 10°, while the CNN–DCNN model with AOA is able to capture the back flow near the edge of airfoil.

Figure 15 shows the effect of including the Reynolds number in the input matrix on the performance of the network model (comparison between case 1 and case 3). The studied cases are trained with the AOA of 6° and Re of [6000, 7000, 8000, 9000, 10 000] and are tested with three different Re = [5500, 8500, 11 000]. It can be clearly seen that the performance of the network model is improved when the information of the Reynolds number is introduced to the network.

In summary, both Figs. 14 and 15 encourage including important physical information when constructing the input matrix; otherwise the performance of the network model can be hurt.

IV. DISCUSSION

The first principle, such as conservation laws, has been the dominant building block for fluid flow modeling over the past centuries. The most prominent model, the Navier–Stokes equations, has been proved to be valid for all linear fluid flow as long as the continuum assumption meets. However, the computational cost is expensive, and the numerical simulation is often too slow for real-time control. Therefore, surrogate models and reduced order models (ROMs) appear, which establish a low-order transfer function of the system, thereby replacing the original full-order partial differential system. The surrogate models and reduced order models can solve the flow field much faster than CFD by learning priorly known high-fidelity data. For decades, the proper orthogonal decomposition (POD) method provides powerful tools for building ROMs for the high-dimensional system by capturing the dynamics of the modes containing highest energy of the system; however, the POD method is also limited because it is a linear combination of eigenvectors and eigenvalues and does not explicitly account for the nonlinear interactions of the highly nonlinear dynamic system.[30]

The dramatic development of machine learning has prompted the invention of various algorithms for dealing with large datasets, which provides new avenues for reduced order modeling of fluid dynamic systems. A machine learning-based model utilizes a large database for iterative training and implicitly expresses physical laws in the form of the model parameters. However, it should be noticed that compared with the CFD simulation, the data-driven neural network model relies on the dataset and indirectly respects certain physical constrains. Hence, the machine learning based data-driven modeling of fluid flow requires the availability of partial prior knowledge of the governing equations and constraints, and the sufficiency of the information of the training dataset about the physical problem is one way ensuring this. Fortunately, with the advance of the computational capability and experimental technique, fluid dynamics has been a data-rich field and thus is amenable to machine learning algorithms. It is worth to point out that there are several approaches attempting to directly encode the physical constraints into the network models, either via the loss function[31] or using the neural network to construct the implicit constitutive equations.[32]

For an unsteady data-driven model, the extrapolative capability in time is always an important and difficult issue. The performance of the model usually deteriorates as time goes as the model becomes more "far away" from the original input, which contains correct/real information. While this is not a serious problem for the current network model, we have used the transient pressure signal on the airfoil surface as the input of the network model, and the information of this pressure signal is always correct (same to real) and would not decay as time goes.

Besides the data, the performance of the machine learning based flow model is also determined by the design of the network architecture. For guiding the design of the network, researchers have tried to find the similarity and connection between the architecture of the neural network and the differential equation. Lu *et al.*[33] bridged the deep neural network design with the numerical differential equations and shows that the network can be interpreted as different numerical discretization. In fact, there are deeper connections between convolutional networks and stencils of numerical schemes, and this have been uncovered recently for developing efficient neural network based partial differential equation (PDE) solvers; some physical interpretation has been put forward and discussed by Mohan[34] and Dong[35,36] recently.

V. CONCLUSIONS

A convolutional neural network based flow model has been developed for fast prediction of the velocity field around an

airfoil. The network model is constituted by convolutional layers and deconvolutional layers. The convolutional layers are designed to capture the intrinsic features directly from the high-dimensional input matrix and represent it in a low-dimensional form. In this article, we design our input matrix that contains the information of the Reynolds number, angle of attack, and time-varying pressure. The convolutional layers are designed to represent the captured low-dimensional features to high-dimensional output. This kind of model is found to be capable of successfully learning and predicting both spatial and temporal characteristics of the vortex shedding. The convolution operation, which is the core of the model, utilizes the characteristics of parameter sharing and deep structures to enhance the robustness and nonlinearity of the model.

The network model was trained to predict the evolution of the flow fields around an airfoil under different flow conditions. The results predicted by the CNN–DCNN model are in good agreement with the flow fields by CFD; the time cost of model training on a single GPU (RTX 2080ti) is about 40 min; furthermore, using the network model, the prediction speedup is about 300 times by CPU and is about 1500 by GPU compared to the CFD simulation. Interestingly, the neural network learned to accurately predict the flow fields that were not included in the training dataset; this implies that the network model has learned the physical mechanism for some certain level. We also illustrated that the performance of the network model could be significantly affected by the construction of the input matrix.

In summary, this is a preliminary attempt to predict the evolution of the flow fields around an airfoil based on the pressure signal on the surface of the airfoil utilizing CNN–DCNN. As the beginning of exploration, the studied flow is kept as two-dimensional and laminar, while the problems of engineering optimization and flow control in real applications are usually three-dimensional and turbulent, which are greatly challenging for the network model. For accurately capturing/modeling the three-dimensional and turbulent flow, the resolution of the CNN kernels, the number of the network layers, the size of the input matrix, and thus the number of the model parameters of the neural network could increase dramatically, which requires higher computational ability and may cause difficulty for model training. Therefore, further efforts need to be made to adequately apply the model to these problems.

ACKNOWLEDGMENTS

This work was supported by Natural Science Foundation of China (Grant No. 11802135), the Fundamental Research Funds for the Central Universities (Grant No. 30919011401), and the Postgraduate Research and Practice Innovation Program of Jiangsu Province (Grant No. KYCX20_0326).

The authors also thank the reviewers for their great suggestions and comments.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]W. Hou, D. Darakananda, and J. D. Eldredge, "Machine learning based detection of flow disturbances using surface pressure measurements," in AIAA Scitech 2019 Forum, 2019.

[2]X. Guo, W. Li, and F. Iorio, "Convolutional neural networks for steady flow approximation," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Journal of the ACM, 2016), pp. 481–490.

[3]S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477–508 (2020).

[4]J. N. Kutz, "Deep learning in fluid dynamics," J. Fluid Mech. **814**, 1–4 (2017).

[5]S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik, "Prediction of aerodynamic flow fields using convolutional neural networks," Comput. Mech. **64**(2), 525–545 (2019).

[6]A. P. Singh, S. Medida, and K. Duraisamy, "Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils," AIAA J. **55**(7), 2215–2227 (2017).

[7]L. Zhu, W. Zhang, J. Kou, and Y. Liu, "Machine learning methods for turbulence modeling in subsonic flows around airfoils," Phys. Fluids **31**(1), 015105 (2019).

[8]V. Sekar, Q. Jiang, C. Shu, and B. C. Khoo, "Fast flow field prediction over airfoils using deep learning approach," Phys. Fluids **31**(5), 057103 (2019).

[9]H. F. S. Lui and W. R. Wolf, "Construction of reduced-order models for fluid flows using deep feedforward neural networks," J. Fluid Mech. **872**, 963–994 (2019).

[10]J. Morton, F. D. Witherden, M. J. Kochenderfer, and A. Jameson, "Deep dynamical modeling and control of unsteady fluid flows," Adv. Neural Inf. Process. Syst. **2018**, 9258–9268; arXiv:1805.07472.

[11]T. P. Miyanawala and R. K. Jaiman, "An efficient deep learning technique for the Navier-Stokes equations: Application to unsteady wake flow dynamics," arXiv:1710.09099 (2017).

[12]J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, "Accelerating Eulerian fluid simulation with convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning* (Proceedings of Machine Learning Research, 2017), Vol. 70, pp. 3424–3433.

[13]N. Omata and S. Shirayama, "A novel method of low-dimensional representation for temporal behavior of flow fields using deep autoencoder," AIP Adv. **9**(1), 015006 (2019).

[14]X. Jin, P. Cheng, W.-L. Chen, and H. Li, "Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder," Phys. Fluids **30**(4), 047105 (2018).

[15]K. Fukami, K. Fukagata, and K. Taira, "Assessment of supervised machine learning methods for fluid flows," Theor. Comput. Fluid Dyn. **34**(4), 497–519 (2020).

[16]M. P. Brenner, J. D. Eldredge, and J. B. Freund, "Perspective on machine learning for advancing fluid mechanics," Phys. Rev. Fluids **4**(10), 100501 (2019).

[17]R. Han, Y. Wang, Y. Zhang, and G. Chen, "A new prediction method of unsteady wake flow by the hybrid deep neural network," arXiv:1908.00294 (2019).

[18]A. J. Newman, *Model Reduction via the Karhunen–Loeve Expansion Part I: An Exposition* (Digital Repository at University of Maryland, 1996).

[19]T. Murata, K. Fukami, and K. Fukagata, "Nonlinear mode decomposition with convolutional neural networks for fluid dynamics," J. Fluid Mech. **882**, A13 (2020).

[20]R. Maulik, O. San, A. Rasheed, and P. Vedula, "Subgrid modelling for two-dimensional turbulence using neural networks," J. Fluid Mech. **858**, 122–144 (2019).

[21]A. J. Holden *et al.*, "Reducing the dimensionality of data with neural networks," Science **313**, 504–507 (2006).

[22]X. Cao, "A practical theory for designing very deep convolutional neural networks," Technical Report 2015.

[23]C. Li *et al.*, AM-LFS: AutoML for Loss Function Search, No. 2, 2019.

[24]J. Bouvrie, Notes on Convolutional Neural Networks (2006), see http://cogprints.org/5869/1/cnn_tutorial.pdf.
[25]N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, and M. Smelyanskiy, "On large-batch training for deep learning: Generalization gap and sharp minima," in 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings (2019), pp. 1–16, arXiv:1609.04836.
[26]E. Yilmaz and B. J. German, "A convolutional neural network approach to training predictors for airfoil performance," in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (AIAA, 2017), pp. 1–19.
[27]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).
[28]S. Lee and D. You, "Data-driven prediction of unsteady flow over a circular cylinder using deep learning," J. Fluid Mech. **879**, 217–254 (2019).
[29]K. Yousefi and A. Razeghi, "Determination of the critical Reynolds number for flow over symmetric NACA airfoils," in *2018 AIAA Aerospace Sciences Meeting* (AIAA, 2018), pp. 1–11.
[30]A. T. Mohan and D. V. Gaitonde, "A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks," arXiv:1804.09269 (2018).
[31]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[32]J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," J. Fluid Mech. **807**, 155–166 (2016).
[33]Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," in *Proceedings of the 35th International Conference on Machine Learning* (ICML, 2018), Vol. 7, pp. 5181–5190.
[34]A. Mohan, D. Daniel, M. Chertkov, and D. Livescu, "Compressed convolutional LSTM: An efficient deep learning framework to model high fidelity 3D turbulence," arXiv:1903.00033 (2019).
[35]Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-net: Learning PDEs from data," in Proceedings of the 35th International Conference on Machine Learning, PMLR (2018), Vol. 80, pp. 3208–3216, arXiv:1710.09668 (2017).
[36]B. Dong, Q. Jiang, and Z. Shen, "Image restoration: Wavelet frame shrinkage, nonlinear evolution PDES, and beyond," Multiscale Model. Simul. **15**(1), 606–660 (2017).