Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 360 (2020) 112766
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Data-driven reduced order model with temporal convolutional neural network

Pin Wu[a,b,*], Junwu Sun[b,*], Xuting Chang[b], Wenjie Zhang[b], Rossella Arcucci[c], Yike Guo[b,c], Christopher C. Pain[c]

[a] *State Key Laboratory of Aerodynamics, China Aerodynamics Research And Development Center, Mianyang Sichuan 621000, China*
[b] *School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China*
[c] *Data Assimilation Lab, Data Science Institute, Imperial College London, London SW7 2AZ, UK*

Received 23 April 2019; received in revised form 13 November 2019; accepted 25 November 2019
Available online 19 December 2019

Abstract

This paper presents a novel model reduction method based on proper orthogonal decomposition and temporal convolutional neural network. The method generates basis functions of the flow field by proper orthogonal decomposition, and the coefficients are taken as the low-dimensional features. Temporal convolutional neural network is used to construct the model for predicting low-dimensional features. In this work, the training data are obtained from high fidelity numerical simulation. Compared with recurrent networks, temporal convolutional neural network is more effective with fewer parameters. The model reduction method developed here depends only on the solution of flow field. The performance of the new reduced order model is evaluated using numerical case: flow past a cylinder. Experimental results illustrate that time cost is reduced by three orders of magnitude, and convolutional architecture is beneficial to construct reduced order model. The speed-up ratio is linear with the computational scale of the numerical simulation.

Keywords: Reduced order model; Proper orthogonal decomposition; Deep learning; Temporal convolutional network

1. Introduction

Numerical simulation has been widely applied in many fields including aerospace engineering [1], environmental sciences [2], bio-medicine [3] and industrial design [4]. It provides powerful technical support for solving industrial problems and making scientific research in these fields. However, high fidelity numerical simulations of complex systems consume vast time and computing resources. Reduced order model (ROM) reduces the computational complexity by using a simplified system instead of the original complex system. It can greatly reduce the computational complexity and save the time cost of numerical simulation while ensuring accuracy. At present, ROM has been successfully applied in computational fluid dynamics [5–8].

ROM is a simplified model used to replace the complex dynamics system. In general, it can be divided into two categories according to the way of constructing ROM. The first category of ROMs is constructed by system

* Corresponding authors at: School of Computer Engineering and Science, Shanghai University, Shanghai, China.
E-mail addresses: wupin@shu.edu.cn (P. Wu), shu2017sjw@shu.edu.cn (J. Sun).

https://doi.org/10.1016/j.cma.2019.112766
0045-7825/

identification method which depends only on inputs and outputs of the source system. Modelling the relationship between inputs and outputs directly, a simple model is constructed to replace the original complex one. The second one is built based on the characteristic modes of flow field. ROM is obtained by mapping the full order system to low-dimensional space constructed by the modes. The second kind can better simulate the details and non-linearity of fluid motion [9]. ROM based on proper orthogonal decomposition (POD) is typical one of it. Since Romanowski [10] applied POD to the construction of reduced order model successfully in 1996, researchers have proposed various model reduction methods based on POD, including intrusive methods and non-intrusive methods. Intrusive ROMs are dependent on the source code of the dynamic system which is difficult to modify and they suffer from instability and nonlinear efficiency issues [11–13]. ROM based on POD and Galerkin projection [14] is a representative intrusive ROM. Non-intrusive ROMs are mainly constructed by combining POD with interpolation methods including Kriging [15], RBF [16,17], Smolyak [18], etc. The predictive ability of them is determined by interpolation functions and the sample data. In recent years, with the development of deep learning, constructing non-intrusive ROMs with neural networks has become a new research hotspot.

Deep learning has excellent performance in mining the latent information in data and has been successfully applied in solving problems in some areas like natural language processing [19], computer vision [20]. Neural networks have tremendous ability to fit function and it can fit almost any unknown function theoretically. That is the ability which makes it possible for neural networks to model complex flows. Deep learning has had an impact on fluid dynamics [21] and there have been related research combining reduced order modelling with deep learning. These researches have demonstrated that the application of deep learning can make the reduced order modelling have better research prospects.

In the procedure of constructing ROMs, deep learning is mainly used for time series modelling which is completed by interpolation methods formerly. Recurrent neural network (RNN), especially long short-term memory (LSTM) [22], has an excellent performance in modelling sequence data. Therefore, RNN is the main deep learning technique used in conjunction with POD to construct reduced order models. Mannarino and Mantegazze [23] developed a reduction scheme based on recurrent neural networks for a nonlinear aerodynamic model. Kani [24] adopted a deep residual recurrent neural network and proposed a new model reduction method for nonlinear dynamic systems. Wang [25] proposed deep learning reduced order model (DLROM) based on POD and LSTM. The DLROM can capture complex fluid flow and has better prediction ability than POD and radial basis function (RBF) based reduced order model. Mohan [26] also studied the effects of LSTM and bidirectional LSTM on ROM and applied them in turbulence control. The performance of bidirectional LSTM is worse than LSTM in this research, and it was caused by overfitting. Besides RNN, other machine learning or deep learning techniques were applied in the construction of ROM as well, for example, Gaussian process regression [27], LS-SVM [28], multi-layer perception [29], feed-forward neural network [30], artificial neural network [31], RBF neural networks [32–34].

Although RNN has advantages in sequence modelling, its internal structure is complicated and it has a large number of parameters. To its opposite is, convolutional neural network (CNN) does not have these sorts of problems because of its own characteristics. CNN has been applied to the prediction in related researches, [35–39]. Bai [40] described a general temporal convolutional network (TCN) to be applied in sequence modelling and made an extensive systematic comparison of convolutional and recurrent architectures on multiple tasks. Compared to recurrent neural networks, TCN architecture is simpler and clearer. Moreover, experimental results indicate that TCN has better performance. Thus, we adopt temporal convolutional architecture and proposed new model reduction method based on it. Moreover, different from previous researches [25,26], the proposed method here does not need to construct models for each POD coefficient. That makes the method more efficient.

In this work, authors have developed a new model reduction method based on POD and temporal convolutional neural network. In this method, the solution snapshots are generated by numerical simulation. POD is employed to generate basis functions optimally representing the problem, and temporal convolutional neural network which is simpler and cleaner than recurrent architecture is used to learn the latent physics dynamics. Four experiments are conducted to explore the performance of the proposed method.

The remainder of this paper is organized as follows. Section 2 introduces some techniques used in the proposed method, which includes dilated convolution and residual connection. Section 3 states the model reduction method based on POD and temporal convolutional neural network step by step. Section 4 is the analysis of experiments, showing the performance of the proposed method. Finally, Section 5 makes a summary of this work.

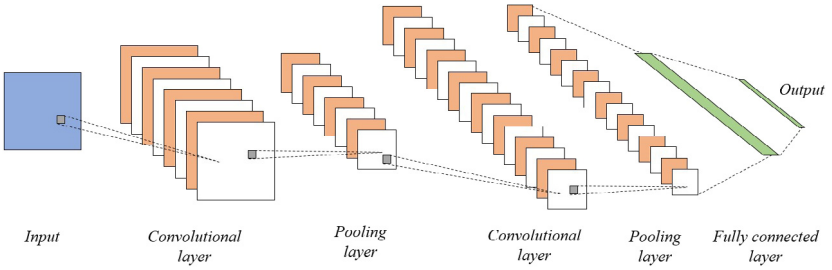


Fig. 1. General architecture of CNN.

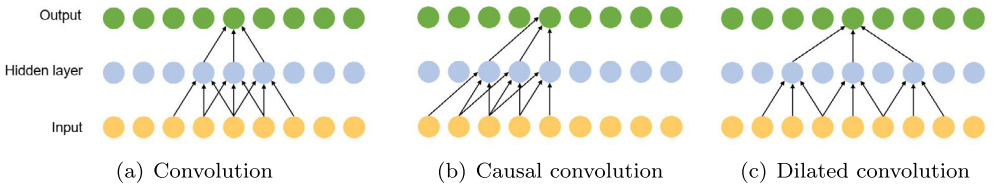


Fig. 2. Three kinds of convolution. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2. Preliminaries

Temporal convolutional network is a variant of convolutional neural network. Commonly, CNNs are composed of convolutional layers, pooling layers, and fully connected layers. Fig. 1 shows a canonical architecture of CNN. In CNN, convolutional layers are responsible for extracting various features from input. Pooling layers are used to reduce the quantity of data and save computing resources. Following them, fully connected layers perform a prediction. As a kind of convolutional neural network, TCN is special in that it uses causal convolution [41], dilated convolution and residual connection [42].

2.1. Causal convolution and dilated convolution

Convolutional layers are the core of convolutional neural networks, extracting various features from input data. One-dimensional convolution can be used to process sequential data. Eq. (1) indicates one-dimensional convolution,

$$O(i) = (I \times K)(i) = \sum_j I(i + j)K(j), \tag{1}$$

where $I(\cdot)$ denotes the input data, $K(\cdot)$ is the one-dimensional convolution kernel and $O(\cdot)$ means the result of the convolution operation. Given one-dimensional sequence, we use convolution, casual convolution and dilated conclusion to convolve sequence respectively. Fig. 2 shows the distinction between them clearly. In Fig. 2, orange means the input, blue the hidden layer, green the output. The most significant difference between them is the data involved. The convolution operates on adjacent elements in accordance with certain rules while causal convolution only operates on elements before the current element. When it comes to time series problems, causal convolution can be considered as an implementation of prediction $p(x_{t+1}|x_t, x_{t-1}, \ldots)$. It is conceivable that when we have to deal with a long time series, a deep network is necessary to make the output have enough receptive field. However, training deep neural network is a difficult task. Dilated convolution is used to figure out the problem. When performing dilated convolution, we need to set a new parameter, dilation rate. Dilation rate defines the distance between elements. In Fig. 2(c), dilation rate of convolution on the input and hidden layer is 1 and 2, respectively. It can be clearly seen from the figure that the receptive field of output in Fig. 2(c) is larger than others.

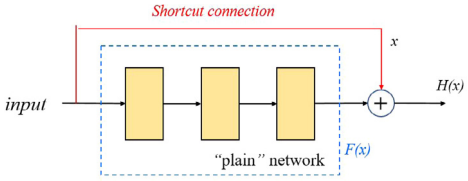


Fig. 3. Residual block.

2.2. Residual connection

In general, we can keep on deepening neural network to make the model have better performance. However, deepening the neural network constantly will bring about two problems, vanishing gradient and network degradation. Due to these problems, it is hard to go on optimizing neural network models. Residual learning framework is proposed in [42] to ease the training of networks. As is shown in Fig. 3, residual structure adds shortcut connection to the "plain" network. Let $H(x)$ be the function that the model need to learn. After adding shortcut connection, $H(x)$ is expressed as $F(x)+x$. Then, the function what neural network needs to fit turns into $F(x)$ which is easier to learn. The residual structure enables the deeper network to work so that the performance of the model can be improved.

3. Reduced order modelling using POD and TCN

In this section, we introduce the model reduction method proposed in this paper. The method we propose to construct reduced order model is data-driven which means it relies only on data. The data can be obtained by high fidelity numerical simulation. For each solution data, e.g., velocity or pressure, the values at the ith moment are stored in the vector $\boldsymbol{V}^{(i)} \in \mathbb{R}^{n\times 1}$, where n is the number of nodes in the computational domain. Then, the set of solutions can be expressed as,

$$\boldsymbol{F} = \{\boldsymbol{V}^{(1)}, \boldsymbol{V}^{(2)}, \ldots, \boldsymbol{V}^{(T)}\}, \boldsymbol{V}^{(t)} = (V_1^{(t)}, V_2^{(t)}, \ldots, V_n^{(t)})^{\top}, 1 \le t \le T, \tag{2}$$

where $\boldsymbol{V}^{(t)}$ represents the data of the flow field at tth moment, n is the number of nodes, T is the last moment.

3.1. Extracting low-dimensional features

The purpose of POD method is to find a sequence of basis functions in the space constructed by snapshots taken at a number of time levels of high fidelity numerical simulation. Snapshot matrix $\boldsymbol{S}$ consists of W solution data sampled from $\boldsymbol{F}$. We obtain the set of basis functions $\boldsymbol{\varphi}_i$ by performing singular value decomposition. Firstly, snapshot matrix subtracts the mean vector of itself,

$$\widetilde{\boldsymbol{S}} = \boldsymbol{S} - \overline{\boldsymbol{S}}, \tag{3}$$

where $\overline{\boldsymbol{S}} \in \mathbb{R}^{n\times 1}$ is the mean vector and n is the node number. Then, performing singular value decomposition,

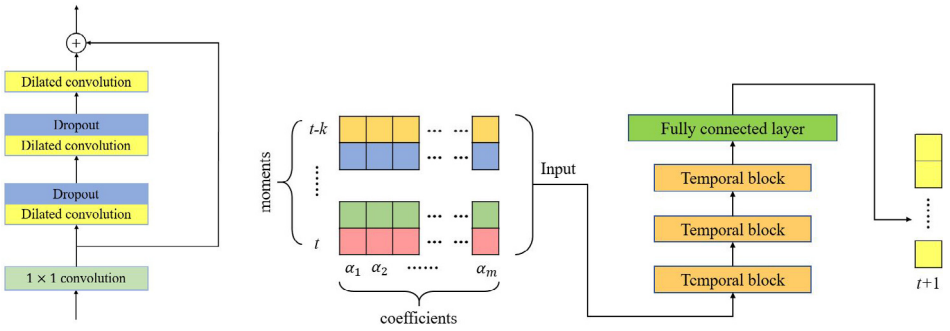
$$\widetilde{\boldsymbol{S}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{B}^{\boldsymbol{T}}. \tag{4}$$

In Eq. (4), $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ is the matrix consisting of the eigenvector of $\widetilde{\boldsymbol{S}}\widetilde{\boldsymbol{S}}^{\top}$, $\boldsymbol{B} \in \mathbb{R}^{W\times W}$ is the eigenvector matrix of $\widetilde{\boldsymbol{S}}^{\top}\widetilde{\boldsymbol{S}}$, and $\Sigma \in \mathbb{R}^{n\times W}$ is a diagonal matrix. The elements on the diagonal of Σ are singular values of $\widetilde{\boldsymbol{S}}$, and the singular values indicate the energy contained in the corresponding basis function. We rank the eigenvalues from large to small and select the first m eigenvalues in turn. Then, we can get the basis functions by Eq. (5),

$$\boldsymbol{\varphi}_i = \widetilde{\boldsymbol{S}}\boldsymbol{B}_i/\sqrt{\lambda_i}, i \in \{1, 2, \ldots, I\}, \tag{5}$$

where λ_i is the singular value. After generating the basis functions, any variable $\boldsymbol{V}$ of the flow field at the tth moment can be expressed as,

$$\boldsymbol{V}^{(t)} = \overline{\boldsymbol{S}} + \sum_{i=0}^{m} \alpha_i^{(t)} \boldsymbol{\varphi}_i, \tag{6}$$



(a) Temporal block (b) General structure of model

Fig. 4. The structure of model.

where $\overline{\boldsymbol{S}}$ is the mean of snapshots, m is the number of selected basis functions, $\boldsymbol{\varphi}_i$ denotes basis function and $\alpha_i^{(t)}$ is the ith component of the corresponding coefficient vector $\boldsymbol{\alpha}^{(t)}$. The energy contained in m eigenvalues can be measured by the following formula,

$$\eta = \frac{\sum_{i=0}^{m} \lambda_i^2}{\sum_{j=0}^{W} \lambda_j^2}, \tag{7}$$

η tends to 1 as m is increased to W, and that means there is no loss. We take the POD coefficient as the low-dimensional feature of the data and use it as the training data of the neural network.

3.2. Modelling the latent physics using TCN

After obtaining POD basis functions, we can build the expression in Eq. (6) for the flow field and take the coefficients as the low-dimensional features of the numerical solution. Then, the prediction problem can be written as

$$\boldsymbol{\alpha}^{(t+1)} = f(\boldsymbol{\alpha}^{(t-k)}, \ldots, \boldsymbol{\alpha}^{(t-2)}, \boldsymbol{\alpha}^{(t-1)}, \boldsymbol{\alpha}^{(t)}). \tag{8}$$

In Eq. (8), $\boldsymbol{\alpha}^{(t)}$ is the low-dimensional feature at time level t, k determines the time span of input. That is to say, previous $k+1$ moments are input into the function. In our work, the function $\widetilde{f}(\boldsymbol{\alpha}^{(t-k)}, \ldots, \boldsymbol{\alpha}^{(t-2)}, \boldsymbol{\alpha}^{(t-1)}, \boldsymbol{\alpha}^{(t)})$, which approximates the function in Eq. (8), is constructed using TCN and k is 5. The structure of temporal convolutional neural network in this work is shown in Fig. 4. The main part of the model is temporal block. A temporal block is shown in Fig. 4(a). It is composed of convolution layers and dropout layers used to prevent overfitting. 1×1 convolution layer is operated at first and it is followed by three dilated convolution layers and two dropout layers. In order to facilitate network training, residual structure is also introduced. Fig. 4(b) shows the neural network. Previous $k+1$ moments data are input into the neural network. Previous $k+1$ moments data are organized as two-dimensional data. Rows of the two-dimensional data represent moments and columns are the corresponding modes. The hyper-parameters in the network, for example, dilation rate, dropout value, convolution kernel size, should be determined by the dimension of input. Finally, a fully connected layer is used to predict the output. The output is the coefficients at $t+1$.

Finally, reduced order model consists of the basis functions and the neural network. Fig. 5 shows the framework of model reduction. In Fig. 5, $\boldsymbol{V}_p^{(i)}$ denotes the ith data taken for POD, $\boldsymbol{\alpha}_r^{(t+1)}$ and $\boldsymbol{V}_r^{(t+1)}$ is the ROM prediction of coefficients and solution at $t+1$th moment. We divide the procedure into two parts: **Stage I** and **Stage II**. The reduced order model is constructed in **Stage I** and data prediction in **Stage II**. In order to clearly describe the process of reduced order modelling method proposed in this paper, we list the algorithms as follows. T_P in Algorithm 2 denotes the number of prediction steps.

Algorithm 1 Stage I: Training model

Input: $\boldsymbol{V}^{(1)},\boldsymbol{V}^{(2)},\cdots,\boldsymbol{V}^{(T)}$, m, W

Output: $\boldsymbol{\Phi}$, $model$ $\widetilde{f}$

1: Choose W solution data to construct snapshot matrix $\boldsymbol{S}$

2: $\widetilde{\boldsymbol{S}} = \boldsymbol{S}$ - $\overline{\boldsymbol{S}}$

3: Extract m POD basis functions, $\boldsymbol{\Phi}=\{\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2,\cdots,\boldsymbol{\varphi}_m\}$

$$\boldsymbol{\Phi} = pod(\widetilde{\boldsymbol{S}})$$

4: **for** $t = 1, 2, \cdots, T$ **do**

5: Calculate corresponding POD coefficients,

$$\boldsymbol{\alpha}^{(t)} = \boldsymbol{\Phi} \cdot (\boldsymbol{V}^{(t)} - \overline{\boldsymbol{S}})$$

6: **end for**

7: Take $\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(2)},\cdots,\boldsymbol{\alpha}^{(T)}$ as training data, train neural network, obtain the model $\widetilde{f}$

$$\boldsymbol{\alpha}^{(t+1)} = \widetilde{f}(\boldsymbol{\alpha}^{(t-k)}, \ldots, \boldsymbol{\alpha}^{(t-2)}, \boldsymbol{\alpha}^{(t-1)}, \boldsymbol{\alpha}^{(t)})$$

Algorithm 2 Stage II: Model prediction

Input: $model$ $\widetilde{f}$,$\boldsymbol{\Phi}$,$\boldsymbol{V}^{(1)}$,$\boldsymbol{V}^{(2)}$,$\boldsymbol{V}^{(3)}$,$\boldsymbol{V}^{(4)}$,$\boldsymbol{V}^{(5)}$,T_P

Output: $\boldsymbol{V}^{(6)},\boldsymbol{V}^{(7)}, \ldots, \boldsymbol{V}^{(5+T_P)}$

1: Calculate coefficient of the initial data

2: **for** $q = 1, 2, \cdots, 5$ **do**

$$\boldsymbol{\alpha}^{(q)} = \boldsymbol{\Phi}\cdot(\boldsymbol{V}^{(q)} - \overline{\boldsymbol{S}})$$

3: **end for**

4: **for** $i = 6, 7, \cdots, 5 + T_P$ **do**

5: (i) Predict POD coefficient at next time step:

$$\boldsymbol{\alpha}^{(i+1)} = \widetilde{f}(\boldsymbol{\alpha}^{(i-k)}, \cdots, \boldsymbol{\alpha}^{(i-2)}, \boldsymbol{\alpha}^{(i-1)}, \boldsymbol{\alpha}^{(i)})$$

6: (ii) Calculate flow data at next time step

$$\boldsymbol{v}^{(i+1)} = \boldsymbol{\Phi}\boldsymbol{\alpha}^{(i+1)} + \overline{\boldsymbol{S}}$$

7: **end for**

4. Experiments

In order to verify the validity of the model reduction method proposed in this paper, experiments based on numerical case, flow past a cylinder, were conducted. *Fluent* provides high fidelity solutions as well as the snapshots for generating POD basis functions. Solutions also provide the training data for neural network. In this work, *Scikit-learn* and *Keras* libraries are used to perform the singular value decomposition and construct neural network. We conduct four experiments to explore the ability of our method. The first experiment is the validation of the proposed method. The second experiment compares the proposed method, LSTM, and gated recurrent unit (GRU) [43]. Research on the effect of the input on prediction is carried out in the third experiment. The last one is about the acceleration effect of reduced order model on numerical simulation with different computational scales.

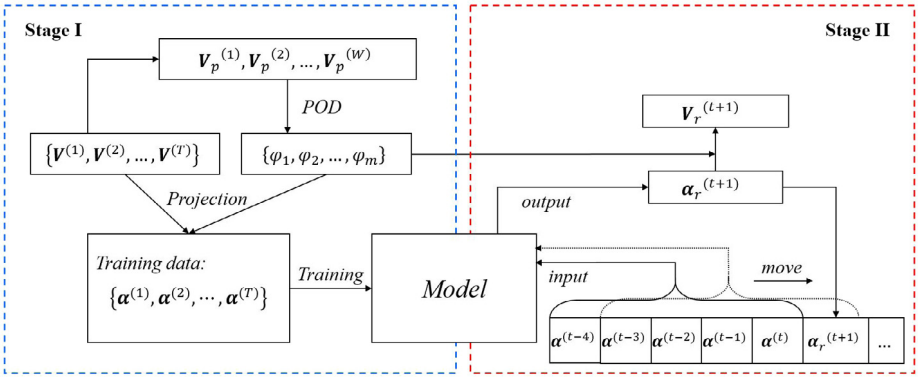
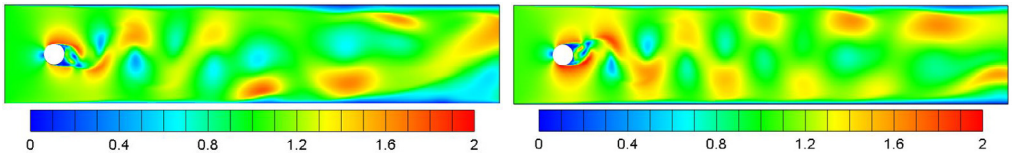


Fig. 5. The framework of our method.



(a) t_1=7.3 second (b) t_2=9.3 second

Fig. 6. Velocity field of flow past a cylinder with 71,580 nodes.

In experiments, we use root mean square error (RMSE) as the criterion to evaluate the accuracy of the reduced order model,

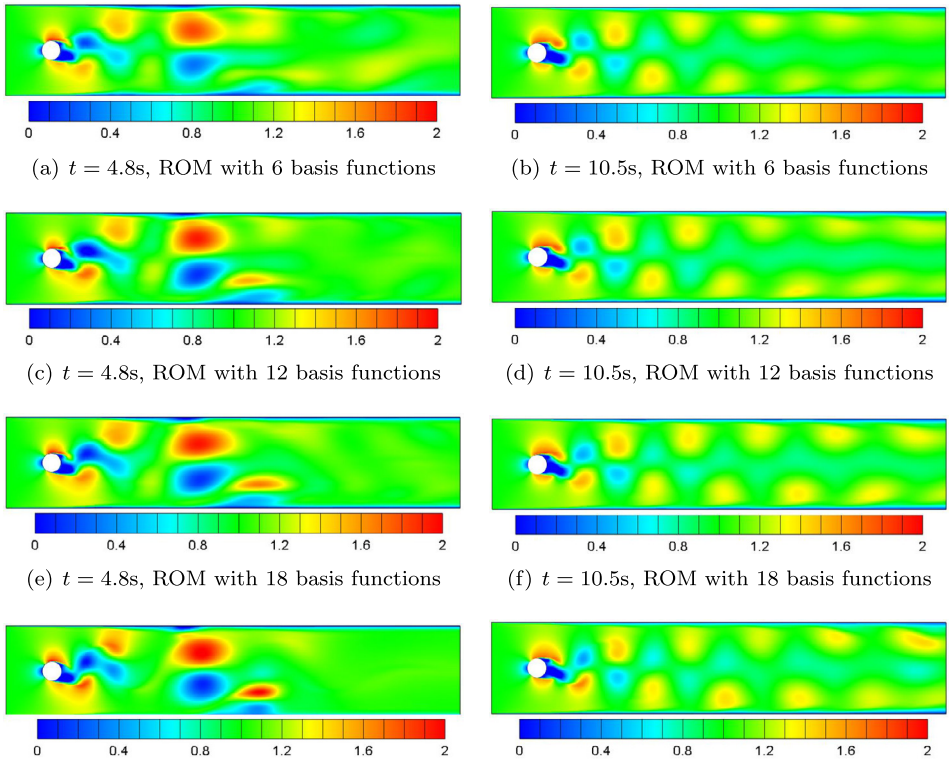
$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (V_i^{(t)} - V_{rom,i}^{(t)})^2}{n}}. \tag{9}$$

where n represents the number of nodes in the computational domain, $V_i^{(t)}$ represents the ith component of the data at tth moment, and $V_{rom,i}^{(t)}$ represents the ith component of the predicted solution of the reduced order model at tth moment. In this work, all predictions are the single-step prediction. All experiments are carried out on a laptop with $Core$ i7-7700H CPU and 16G memory. The input in this work contains data at previous five moments.

4.1. Reduced order models of flow past a cylinder

We perform a reduced order modelling experiment based on numerical case, flow past a cylinder, to verify the validity of the proposed method. Flow past a cylinder is simulated by *Fluent*, and the mesh is generated by *ICEM*. The computational domain of the simulation is 5 m long and 1 m wide. The coordinate of the centre of the cylinder is (0.5, 0.5) in the flow field, the radius of the cylinder is 0.1 m, and the fluid flows from left boundary to the right at 1 meter per second. The density of fluid is 1 kilogram per cubic meter and time step is 0.01 s. By the software $ICEM$, we divided the computational domain into 71,580 nodes to ensure the accuracy of the simulation. Fig. 6 shows the velocity of flow field at $t_1 = 7.3$ s, $t_2 = 9.3$ s. In this experiment, we build a reduced order model only for the velocity, however, the method proposed is applicable to any variables.

In this work, to prove the effectiveness of the proposed method, authors choose three different numbers of basis functions, 6, 12, 18, respectively. The snapshots are constructed by the solution data from 0 s to 12 s. 80% of the data are used as training data, while the remaining 20% are used for validation/test. In this part, we employ the proposed method flexibly to the construction of ROM. Concretely, the structure of network consists of three temporal blocks and one fully connected layer in this part. The dilation rates are (1,1), (1,2), (2,5) orderly in the temporal block. The kernel size is 3×3. The dropout value is set as 0.3. The epoch is 50, and batch size is 32. Optimizer of the neural network is $Adam$ [44]. Loss function is mean squared error. We normalize the data so that



(a) $t = 4.8$s, ROM with 6 basis functions (b) $t = 10.5$s, ROM with 6 basis functions

(c) $t = 4.8$s, ROM with 12 basis functions (d) $t = 10.5$s, ROM with 12 basis functions

(e) $t = 4.8$s, ROM with 18 basis functions (f) $t = 10.5$s, ROM with 18 basis functions

(g) $t = 4.8$s, numerical solution (h) $t = 10.5$s, numerical solution

Fig. 7. Prediction of velocity field.

the range of the data is $[-1, 1]$. Correspondingly, the activation function of the output layer is $tanh$. Activation functions adopted in the convolutional layers are all $Relu$.

Figs. 7 and 8 show the predictive ability of the proposed method together. Fig. 7 presents the forecasting result of flow field at two moments, $t_1 = 4.8$ s, $t_2 = 10.5$ s. Obviously, we can observe that ROM with 18 basis functions achieves the best performance because of that more basis functions contain more energy and the error caused by POD is smaller. Fig. 8 is the RMSE between the numerical solution and ROM solution. The data in Fig. 8 consist of two parts. First half of the data is in the training set and the latter half is in the test set. The error in Fig. 8 goes down because the flow field tends to be stable and the prediction becomes easier. Figs. 7 and 8 indicate that our method can predict the flow field well.

4.2. Comparison of convolution and recurrent architecture

As mentioned above, recurrent architecture is good at modelling sequence data, but its internal architecture is complicated and it has many parameters. Meanwhile, convolutional architecture is simpler and cleaner than recurrent architecture. It makes the convolutional architecture be more suitable for modelling complex sequence data. Thus, we compare the two architectures to prove the merit of our method.

The experiment is conducted to show that convolutional architecture is simpler while it has higher precision. We compare convolutional architecture and recurrent architecture. We apply two representative recurrent architectures, LSTM and GRU, in the construction of ROM. In this part, all ROMs are based on 18 basis functions. Both recurrent networks have the same structure, four hidden layers, and each hidden layer has 128 cells. The inputs to all networks are the same. Moreover, the values of epoch and batch size are the same as well. Similarly, dropout is also introduced into the recurrent network to prevent overfitting. Fig. 9 and Table 1 show the experimental results. Fig. 9 is the RMSE between numerical solution and predictions of three networks. Same as the previous part, the result shown

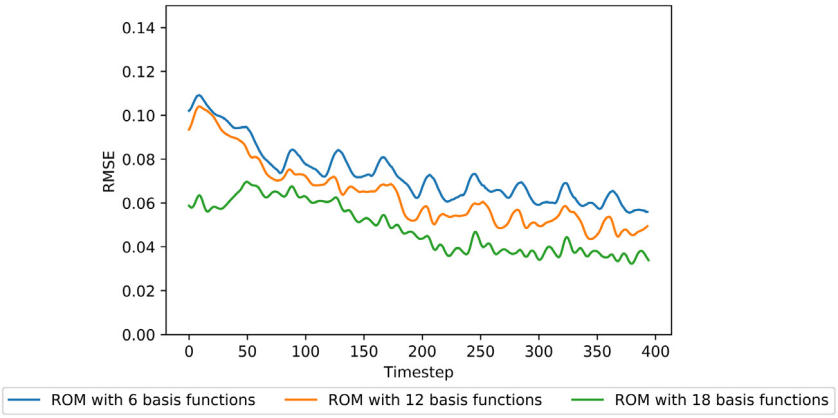


Fig. 8. RMSE between numerical solution and ROM prediction.

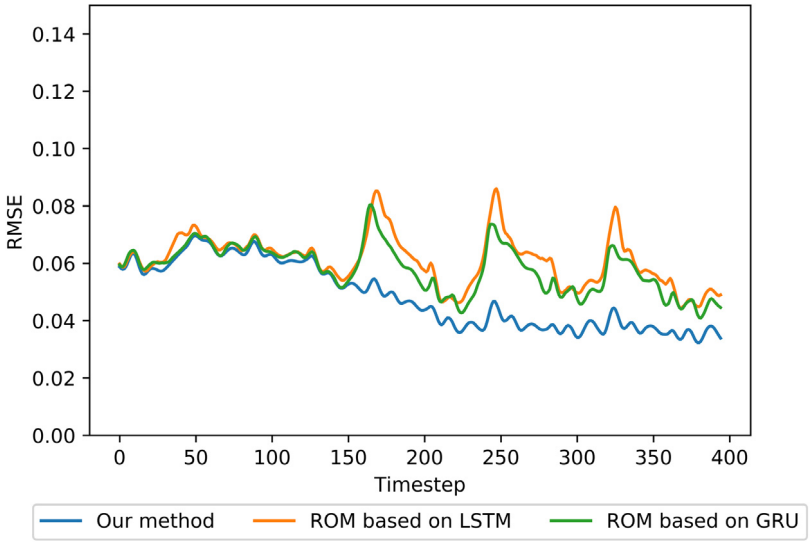


Fig. 9. RMSE of ROMs based on different deep learning techniques.

Table 1
Comparison of convolutional and recurrent architectures.

	Our method	GRU	LSTM
Number of parameters	87,018	354,834	472,338
Average RMSE	0.0479	0.0523	0.0535

in Fig. 9 consists of two parts: training set and test set. The boundary is around the 150th timestep. In training set, of course, all three methods can fit the data and make a great prediction. However, it changes in the test set. Convolutional architecture achieves better performance and is more stable. Besides, the numbers of parameters to be trained are shown in Table 1. Table 1 also presents the average value of RMSE of three methods on training and test sets. It reveals that our method has five times fewer parameters than LSTM and three times fewer than GRU approximately. In summary, our method which is a convolutional architecture has the best performance with the least parameters. Values in Table 1 are all dimensionless.

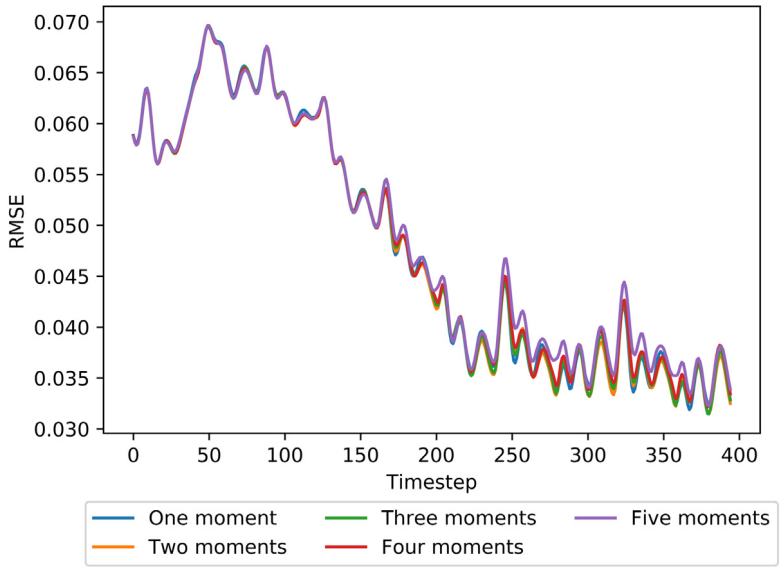


Fig. 10. RMSE of ROMs with different inputs.

Table 2
Comparison of models with different inputs.

	TCN-1	TCN-2	TCN-3	TCN-4	TCN-5
Average RMSE	0.04788	0.04775	0.04780	0.04782	0.04798

4.3. Effect of the quantity of input on prediction

Another interesting part is the effect of the number of inputs on the prediction. We take the previous five moments data as the input in the previous two subsections. The reason is that it can provide abundant information for the prediction. However, an experiment to prove its correctness is necessary.

In this part, five models with different inputs are trained. They take data of previous one, two, three, four, five moments as the input, respectively. Except for the input, they have same structure and hyper-parameters. Fig. 10 and Table 2 indicate the relation between the input and model performance. The temporal interval is the same as previous two subsections. Actually, nothing useful can be obtained by Fig. 10 because these lines almost overlap. The distinction between them is very tiny. Therefore, we provide the average RMSE of five models, that is Table 2. TCN-j means that the model takes previous j moments data as the input. Seen from the table, the five models get similar performance. TCN-2 predicts slightly better than other models. It can be inferred that too many inputs are of no use for the prediction because of overfitting. Too many inputs make the function to be modelled complicated. In practice, carrying out experiments to find the approximate quantity is necessary.

4.4. Relationship between speed-up ratio and computational scale

We investigate the relationship between the number of nodes in the computational domain and speed-up ratio. The speed-up ratio is defined by Eq. (10),

$$Sr = \frac{t_n}{t_r}. \tag{10}$$

Sr denotes the speed-up ratio, t_n is the time for numerical simulation and t_r is the time for prediction. The time for training the neural network and POD is not contained. Based on Section 4.1, we still use $ICEM$ to generate four other meshes with different sizes. The number of nodes is $n_1 = 19750$, $n_2 = 53605$, $n_3 = 142285$, $n_4 = 203900$ respectively. The boundary conditions and initial conditions of these four simulations are the same

Table 3
Comparison of speed-up ratio.

Number of nodes	Numerical simulation (s)	Training (s)	Prediction (s)	Speed-up ratio
19 750	0.7961	68.9333	0.00246	323.61
53 605	1.5086	72.6645	0.00251	601.03
71 580	1.9597	74.8104	0.00245	799.87
143 480	4.5623	77.5268	0.00244	1869.79
209 300	5.5436	80.9682	0.00209	2652.44

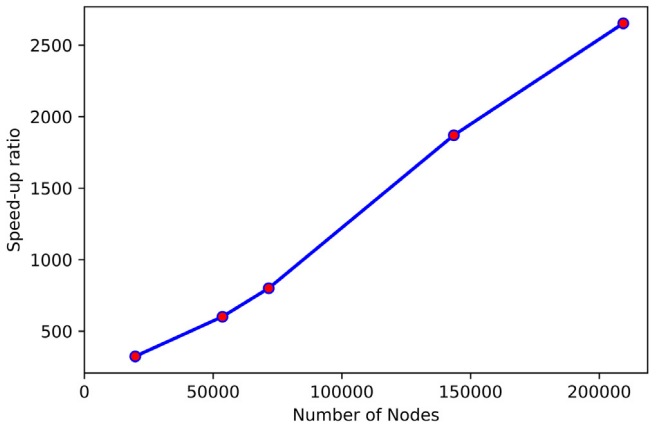


Fig. 11. Relationship between number of nodes and speed-up ratio.

as it in Section 4.1. We extract 18 POD basis functions for each numerical simulation in this part. And the hyper-parameters of neural networks are also the same.

The main function and aim of reduced order model are to save computing resources and elevate the efficiency. Table 3 shows the comparison of the speed-up ratio for one snapshot, including the time consumption of numerical simulation, training the TCN and predicting the solution with the trained neural network. The values in Table 3 are all the average of corresponding variables. We record the numerical computation time of one hundred time steps. It is divided by 100, then the average value is obtained. We take the same approach to calculate the time cost of predicting one snapshot. In Table 3, the comparison of speed-up ratio is also presented. Furthermore, in order to visualize the result, Fig. 11 is made. It is the line chart of the speed-up ratio in Table 3. The x-axis represents the number of nodes, and the y-axis is the speed-up ratio. It reveals that with the growth of the number of nodes in computational domain, the speed-up ratio is greater. It can be found the speed-up ratio is approximately linear with number of nodes.

5. Conclusion

This paper proposes a new non-intrusive reduced order model based on POD and temporal convolutional neural network. The model employs POD to extract low-dimensional features from high fidelity numerical solution and applies temporal convolutional neural network to model low-dimensional features. In order to verify the validity of the proposed model reduction method, we performed reduced order modelling experiments based on numerical simulation. The convolutional and recurrent architecture are compared and the result makes it clear that convolutional architecture is more effective in model reduction. Experiments also reveal that appropriate amount of input is important. In addition, we carried out model reduction experiments on the flow around a cylinder with five different meshes, which shows that the speed-up ratio will be greater as the scale of computation continues to increase. Furthermore, the relationship between them is approximately linear.

Deep learning does have excellent performance in data modelling, which provides a new impetus for the research and development of reduced order models. However, training a great neural network needs mass data as training set. Like other areas, for example, natural language processing, computer vision, comprehensive public datasets are

necessary for the research on ROMs based on deep learning. To the best of authors' knowledge, there is one public dataset in the computational fluid dynamics and it is mainly related to turbulence [45]. Moreover, some other techniques are needed to keep the reduced order model effective permanently, like data assimilation. Otherwise, reduced order models will be disabled after predicting recursively due to the error accumulation.

Acknowledgements

The authors would like to thank the reviewers for their valuable comments and effort to improve the manuscript. This work is supported by State Key Laboratory of Aerodynamics, China Aerodynamics Research and Development Center under Grant SKLA20180303, Natural Science Foundation of Shanghai under Grant 19ZR1417700.

References

[1] X. Liu, J. Guo, C. Bai, X. Sun, R. Mou, Drop test and crash simulation of a civil airplane fuselage section, Chin. J. Aeronaut. 28 (2) (2015) 447–456.
[2] Y. Tominaga, T. Stathopoulos, Cfd simulation of near-field pollutant dispersion in the urban environment: A review of current modeling techniques, Atmos. Environ. 79 (2013) 716–730.
[3] K. Valen-Sendstad, K.-A. Mardal, D.A. Steinman, High-resolution cfd detects high-frequency velocity fluctuations in bifurcation, but not sidewall, aneurysms, J. Biomech. 46 (2) (2013) 402–407.
[4] J. Zhang, J.-j. Li, H.-q. Tian, G.-j. Gao, J. Sheridan, Impact of ground and wheel boundary conditions on numerical simulation of the high-speed train aerodynamic performance, J. Fluids Struct. 61 (2016) 249–261.
[5] P. Persson, K. Persson, G. Sandberg, Reduced order modelling of liquid-filled pipe systems, J. Fluids Struct. 61 (2016) 205–217.
[6] X. Chen, Z. Qiu, X. Wang, Y. Li, R. Wang, Uncertain reduced-order modeling for unsteady aerodynamics with interval parameters and its application on robust flutter boundary prediction, Aerosp. Sci. Technol. 71 (2017) 214–230.
[7] D. Xiao, P. Yang, F. Fang, J. Xiang, C.C. Pain, I. Navon, M. Chen, A non-intrusive reduced-order model for compressible fluid and fractured solid coupling and its application to blasting, J. Comput. Phys. 330 (2017) 221–244.
[8] D. Xiao, P. Yang, F. Fang, J. Xiang, C.C. Pain, I.M. Navon, Non-intrusive reduced order modelling of fluid–structure interactions, Comput. Methods Appl. Mech. Engrg. 303 (2016) 35–54.
[9] G. Chen, Y. Li, Advances and prospects of the reduced order model for unsteady flow and its application, Adv. Mech. 41 (6) (2011) 686–701.
[10] M. Romanowski, Reduced order unsteady aerodynamic and aeroelastic models using karhunen-loeve eigenmodes, in: 6th Symposium on Multidisciplinary Analysis and Optimization, 1996, p. 3981.
[11] D. Amsallem, C. Farhat, Stabilization of projection-based reduced-order models, Internat. J. Numer. Methods Engrg. 91 (4) (2012) 358–377.
[12] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (5) (2010) 2737–2764.
[13] H. Chen, et al., Blackbox stencil interpolation method for model reduction (Ph.D. thesis), Massachusetts Institute of Technology, 2012.
[14] D.J. Lucia, P.S. Beran, Projection methods for reduced order models of compressible flows, J. Comput. Phys. 188 (1) (2003) 252–280.
[15] X. Chen, L. Liu, Z. Yue, Reduced order aerothermodynamic modeling research for hypersonic vehicles based on proper orthogonal decomposition and surrogate method, Acta Aeronaut. Astronaut. Sinica 36 (2) (2015) 462–472.
[16] D. Xiao, F. Fang, C. Pain, G. Hu, Non-intrusive reduced-order modelling of the navier–stokes equations based on rbf interpolation, Internat. J. Numer. Methods Fluids 79 (11) (2015) 580–595.
[17] C. Audouze, F. De Vuyst, P.B. Nair, Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations, Numer. Methods Partial Differential Equations 29 (5) (2013) 1587–1628.
[18] D. Xiao, F. Fang, A. Buchan, C. Pain, I. Navon, A. Muggeridge, Non-intrusive reduced order modelling of the navier–stokes equations, Comput. Methods Appl. Mech. Engrg. 293 (2015) 522–541.
[19] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing, IEEE Comput. Intell. Mag. 13 (3) (2018) 55–75.
[20] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, Deep learning for computer vision: A brief review, Comput. Intell. Neurosci. (2018).
[21] J.N. Kutz, Deep learning in fluid dynamics, J. Fluid Mech. 814 (2017) 1–4.
[22] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[23] A. Mannarino, P. Mantegazza, Nonlinear aeroelastic reduced order modeling by recurrent neural networks, J. Fluids Struct. 48 (2014) 103–121.
[24] J.N. Kani, A.H. Elsheikh, Dr-rnn: A deep residual recurrent neural network for model reduction, arXiv preprint arXiv:1709.00939.
[25] Z. Wang, D. Xiao, F. Fang, R. Govindan, C.C. Pain, Y. Guo, Model identification of reduced order fluid dynamics systems using deep learning, Internat. J. Numer. Methods Fluids 86 (4) (2018) 255–268.
[26] A.T. Mohan, D.V. Gaitonde, A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks, arXiv preprint arXiv:1804.09269.
[27] M. Guo, J.S. Hesthaven, Reduced order modeling for nonlinear structural analysis using gaussian process regression, Comput. Methods Appl. Mech. Engrg. 341 (2018) 807–826.

[28] Z. Chen, Y. Zhao, R. Huang, Parametric reduced-order modeling of unsteady aerodynamics for hypersonic vehicles, Aerosp. Sci. Technol. 87 (2019) 1–14.
[29] J.S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, J. Comput. Phys. 363 (2018) 55–78.
[30] Q. Wang, J.S. Hesthaven, D. Ray, Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem, J. Comput. Phys. 384 (2019) 289–307.
[31] O. San, R. Maulik, Machine learning closures for model order reduction of thermal fluids, Appl. Math. Model. 60 (2018) 681–710.
[32] M. Winter, C. Breitsamter, Reduced-order modeling of unsteady aerodynamic loads using radial basis function neural networks, Dtsch. Ges. Luft- Raumfahrt-Lilienthal-Oberth eV (2014).
[33] J. Kou, W. Zhang, Reduced-order modeling for nonlinear aeroelasticity with varying mach numbers, J. Aerosp. Eng. 31 (6) (2018) 04018105.
[34] J. Kou, W. Zhang, Multi-kernel neural networks for nonlinear unsteady aerodynamic reduced-order modeling, Aerosp. Sci. Technol. 67 (2017) 309–326.
[35] H. Li, O.L. Kafka, J. Gao, C. Yu, Y. Nie, L. Zhang, M. Tajdari, S. Tang, X. Guo, G. Li, et al., Clustering discretization methods for generation of material performance databases in machine learning and design optimization, Comput. Mech. (2019) 1–25.
[36] X. Hui, Z. Yuan, J. Bai, Y. Zhang, G. Chen, A method of unsteady periodic flow field prediction based on the deep learning, Acta Aerodyn. Sinica 37 (3) (2019) 462–469.
[37] H. Chen, W. Qian, L. He, Aerodynamic coefficient prediction of airfoils based on deep learning, Acta Aerodyn. Sinica 36 (2) (2018) 294–299.
[38] Y. Zhang, W.J. Sung, D.N. Mavris, Application of convolutional neural network to predict airfoil lift coefficient, in: 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2018, p. 1903.
[39] V. Sekar, Q. Jiang, C. Shu, B.C. Khoo, Fast flow field prediction over airfoils using deep learning approach, Phys. Fluids 31 (5) (2019) 057103.
[40] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, arXiv preprint arXiv:1803.01271.
[41] A.V.D. Oord, S. Dieleman, H. Zen, K. Simonyan, K. Kavukcuoglu, Wavenet: A generative model for raw audio.
[42] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[43] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder–decoder approaches, arXiv preprint arXiv:1409.1259.
[44] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.
[45] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, G. Eyink, A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence, J. Turbul. (9) (2008) N31.