

Noname manuscript No.
 (will be inserted by the editor)

Machine-learning-based reduced order modeling for unsteady flows around bluff bodies of various shapes

Kazuto Hasegawa

Kai Fukami

Takaaki Murata

Koji Fukagata

Received: March 18, 2020 / Accepted: ???

Abstract We propose a method to construct a reduced order model with machine learning for unsteady flows. The present machine-learned reduced order model (ML-ROM) is constructed by combining a convolutional neural network autoencoder (CNN-AE) and a long short-term memory (LSTM), which are trained in a sequential manner. First, the CNN-AE is trained using direct numerical simulation (DNS) data so as to map the high-dimensional flow data into low-dimensional latent space. Then, the LSTM is utilized to establish a temporal prediction system for the low-dimensionalized vectors obtained by CNN-AE. As a test case, we consider flows around a bluff body whose shape is defined using a combination of trigonometric functions with random amplitudes. The present ML-ROMs are trained on a set of 80 bluff body shapes and tested on a different set of 20 bluff body shapes not used for training, with both training and test shapes chosen from the same random distribution. The flow fields are confirmed to be well reproduced by the present ML-ROM in terms of various statistics. We also focus on the influence of two main parameters: (1) the latent vector size in the CNN-AE, and (2) the time step size between the mapped vectors used for the LSTM. The present results show that the ML-ROM works well even for unseen shapes of bluff bodies when these parameters are properly chosen, which implies great potential for the present type of ML-ROM to be applied to more complex flows.

Keywords Reduced order modeling, machine learning, unsteady wake

1 Introduction

In recent years, a huge amount of detailed flow field information has been accumulated as *fluid big data* thanks to high-resolution numerical simulations and image-based measurements. Understanding essential phenomena and controlling flows based on such big data — as they are — are difficult due to their complexity. Therefore, reduced order models (ROMs) have been utilized as one way to tackle such problems. One of the beauties of ROMs is that they can map a flow field with high dimensions into a low-dimensional space [1]. Lumley [2] introduced the proper orthogonal decomposition (POD), which can express a flow field with several principal modes and the corresponding eigenvalues. Schmid [3] proposed the dynamic mode decomposition (DMD) that extracts the information from flow fields by focusing on a specific frequency. These ROMs are considered to have deepened our understanding [4] and enabled us to control flow phenomena at low computational costs [5]. Despite the great advantages of these linear methods, an annoying problem may be that the number of

Kazuto Hasegawa

Department of Mechanical Engineering, Keio University, Yokohama 223-8522, Japan

Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano, Milano 20156, Italy

Kai Fukami, Takaaki Murata, Koji Fukagata

Department of Mechanical Engineering, Keio University, Yokohama 223-8522, Japan

E-mail: fukagata@mech.keio.ac.jp

modes required to represent a flow often becomes too large to handle because nonlinear phenomena must be approximated by a linear superposition of orthogonal modes. Even for a turbulent channel flow at a low Reynolds number, for instance, 7260 POD modes are required to reconstruct 95% of its total energy [6]. In order to reduce the number of modes, use of the novel nonlinear dimension reduction technique that brought innovation to image recognition [7], i.e., machine learning, can be considered as a good candidate.

In recent years, machine learning techniques, which can automatically extract key features from tremendous amount of data, have achieved noteworthy results in various fields including fluid dynamics owing to the advances in the algorithms centering on *deep learning* [8, 9, 10, 11, 12, 13], which has been enabled by the recent development of computational power. For instance, Ling et al. [14] proposed a tensor basis neural network to predict the Reynolds stress anisotropy tensor for Reynolds-averaged Navier–Stokes simulations. The proposed method was applied to the duct and wavy flows and it showed substantial merits over the conventional eddy viscosity models. Fukami et al. [15] utilized convolutional neural networks (CNN) [16] for a super-resolution reconstruction of two-dimensional turbulence and reported that the customized CNN model can recover the maximum wavenumbers of energy spectrum from grossly coarse low-resolution flow data. A machine learning method was also applied to the flow around a circular cylinder so as to predict the flow fields at various Reynolds numbers from the pressure drag coefficient distribution [17]. Moreover, Viquerat and Hachem [18] have proposed a CNN based method to predict drag coefficients in a two-dimensional low Reynolds number flow around various random shapes generated by Bézier curves. In this way, capability of machine learning has been demonstrated for different kinds of fluid dynamics problems, although it should be noted that the literature on this topic is vast and many other applications exist despite the references provided here.

Of particular interest concerning the machine learning for fluid dynamics is its applications to nonlinear reduced order modeling. San & Maulik [19] proposed an ROM for quasistationary geophysical turbulent flows based on the extreme learning machine. Srinivasan et al. [20] proposed a machine learning model based on a multi-layer perceptron and a long short-term memory (LSTM) [22] to successfully predict temporal behaviors of the coefficients in the nine-equation turbulent flow model. More recently, Murata et al. [21] have proposed nonlinear mode decomposition via CNN autoencoder (CNN-AE) and reported its great advantage over POD for the flow around a circular cylinder and its transient process in terms of the feature extraction of flow fields in lower dimensions.

The objective of the present study is to propose a method of reduced order modeling using CNN-AE and LSTM, which have been separately shown to have great potentials as introduced above. The machine-learned reduced order model (ML-ROM) proposed here is constructed by combining a CNN-AE and an LSTM, which are trained in a sequential manner. The CNN-AE part is trained first to map the high-dimensional flow field obtained by direct numerical simulation (DNS) into a low-dimensional latent space. Then, the LSTM part is trained to predict the temporal evolution of the low-dimensionalized vectors obtained by the CNN-AE. As a test case, we consider two-dimensional unsteady flows around a bluff body. We randomly define the shapes of bluff bodies in order to assess the performance of the present ML-ROM for unseen data. Moreover, the effects of the two key parameters are examined to unveil their influence on the model performance.

The remainder of the paper is organized as follows. Section 2 introduces the details of the training data and the theory of the machine learning models. The results and case studies on the prediction of flows around bluff bodies of various shapes are presented and discussed in section 3. Finally, the concluding remarks are provided in section 4.

2 Methods

2.1 Training data

Two-dimensional direct numerical simulation (DNS) of flows around various bluff bodies, whose shapes are defined randomly, are performed to obtain the flow fields used for training, validation, and assessment of the

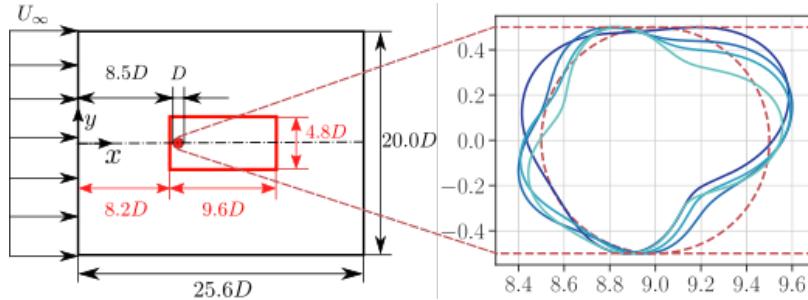


Fig. 1 The computational domain used for DNS (black lines), its subdomain used for the machine learning (red lines), and the shapes of the bluff bodies. Blue lines indicate the example of the bluff bodies defined randomly following equations (3) and (4).

ML-ROM. The governing equations are the incompressible continuity and Navier–Stokes equations, i.e.,

$$\nabla \cdot \mathbf{u} = 0, \quad (1)$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\nabla p + \frac{1}{\text{Re}_D} \nabla^2 \mathbf{u}, \quad (2)$$

where $\mathbf{u} = [u, v]^T$, p , and t denote the velocity, pressure, and time, respectively. All variables are made dimensionless by the fluid density ρ^* , the uniform velocity U_∞^* , and the frontal length D^* of the body, where the superscript * represents dimensional variables. The Reynolds number is set to $\text{Re}_D = U_\infty^* D^* / \nu^* = 100$, where ν^* is the kinematic viscosity.

The computational domain is shown in the left part of figure 1. The center of the bluff body is located $9D$ from the inflow boundary. The uniform velocity $U_\infty = 1$ is given at the inflow boundary, the convective boundary condition is used at the outflow boundary, and the free-slip condition is imposed on the top and bottom boundaries.

The present DNS code is basically the same as that used by Anzai et al. [23] for flows around a square cylinder, except that a ghost-cell method [24] is used to satisfy the no-slip boundary condition on the bluff body surface. The spatial discretization is done by using the energy-conservative second-order finite difference method on a staggered grid system [25], which is uniform in both streamwise (x) and transverse (y) directions with the grid size $\Delta x = \Delta y = 0.025$. The number of computational cells is $(N_x, N_y) = (1024, 800)$. The time integration is done using the low-storage third-order Runge-Kutta/Crank-Nicolson (RK3/CN) scheme [26] with a velocity-pressure coupling similar to the simplified marker and cell (SMAC) method [25]. The time step is set to $\Delta t = 2.5 \times 10^{-3}$. The pressure Poisson equation is solved by means of the fast Fourier transform (FFT) in x direction with the mirroring technique [27] and the tridiagonal matrix algorithm (TDMA) in (y) direction. We have verified for some selected cases that the present grid resolution is sufficiently fine, and we have validated that the time-averaged drag and rms lift coefficients as well as the Strouhal number computed for a circular cylinder (for which references are available) are in good agreement with the references.

As mentioned above, the flows around bluff bodies with various shapes are considered in order to examine whether we can construct a single ML-ROM approximating the function \mathcal{F} corresponding to the time-discretized Navier–Stokes equation $\mathbf{q}^{(n+1)\Delta t} = \mathcal{F}(\mathbf{q}^{n\Delta t})$ (where $\mathbf{q} = [u, v, p]^T$ and the superscript denotes time), which is valid even for unseen shapes. The shape of a single bluff body is defined as

$$r = 0.5 + \sum_{n=1}^4 a_n \sin n\theta + \sum_{n=1}^4 a_{n+4} \cos n\theta, \quad (3)$$

$$\sum_{n=1}^8 a_n = 0.5, \quad (4)$$

where r is the distance between the center and the surface, θ represents the angle from the inflow (i.e., x) direction, and a_n denotes random numbers normalized to satisfy equation (4). The bluff body shapes

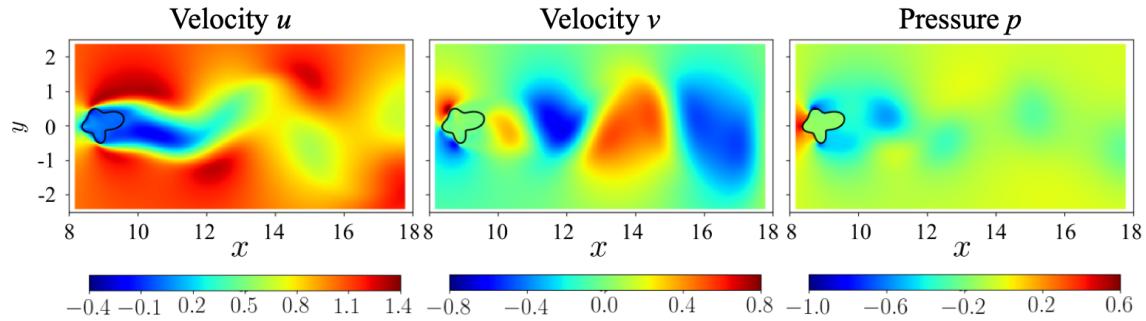


Fig. 2 An example of the velocity and pressure fields around the randomly defined bluff body.

generated using equations (3) and (4) are rescaled so that the frontal length becomes unity and $\text{Re}_D = 100$ in all cases. Fifty different shapes are defined, and the flows around them are produced using the DNS. Moreover, the flow fields are rotated around the x -axis symmetry to increase the amount of training data. In this way, hundred kinds of flows are prepared as the data sets. Note in passing that the achievable range of shapes generated using equations (3) and (4) is limited, and the use of this formulation is intended to be a proof of concept.

In order to focus on the flow around the bluff body, the velocities and pressure (u, v, p) in the region enclosed by the red line in figure 1 are extracted to use for machine learning. The size of the instantaneous field data used for ML-ROM construction is $(\hat{N}_x, \hat{N}_y, N_\phi) = (384, 192, 3)$, where ϕ represents the considered physical quantities. An example of the flow fields is shown in figure 2. In this study, we do not apply any data pre-processing such as normalization or standardization since the order of magnitude is unity for all the quantities thanks to the nondimensionalization, and the bluff body shapes are adjusted to have the equal frontal length (i.e., unity) as mentioned above.

2.2 Machine learning

2.2.1 Convolutional neural network autoencoder (CNN-AE)

The convolutional neural network (CNN) [16] has been widely used in the field of image recognition, and it has also been applied to fluid dynamics in recent years [15, 17, 28] due to its ability to deal with spatially coherent information. The CNN is formed by connecting two kinds of layers: convolution layers and sampling layers.

The convolutional operation performed in the convolution layer can be expressed as

$$s_{ijm} = \sum_{k=0}^{K-1} \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} z_{i+p,j+q,k} W_{pqkm} + b_m, \quad (5)$$

where z_{ijk} is the input value at point (i, j, k) , W_{pqkm} denotes the weight at point (p, q, k) in the m -th filter, b_m represents the bias of the m -th filter, and s_{ijm} is the output of the convolution layer. The schematics of the convolutional operation and a convolution layer without bias are shown in figures 3(a) and (b), respectively. The input is a three-dimensional matrix with the size of $L_1 \times L_2 \times K$, where L_1 , L_2 , and K are the height, the width, and the number of channels (e.g., $K = 3$ for RGB images), respectively. There are M filters with the length H and the K channels. After passing the convolution layer, an activation function $f(\cdot)$ is applied to s_{ijm} , i.e.,

$$z_{ijm} = f(s_{ijm}). \quad (6)$$

Usually, nonlinear monotonic functions are used as the activation function $f(\cdot)$. The sampling layer performs compression or extension procedures with respect to the input data. Here, we use a max pooling operation

for the pooling layer, as summarized in figure 3(c). Through the max pooling operation, the machine learning model is able to obtain the robustness against rotation or translation of the images. In contrast, in the convolutional neural network autoencoder [29] (CNN-AE) explained below, the upsampling layer in the decoder part copies the values of the low-dimensional images into a high-dimensional field, i.e., the nearest neighbour interpolation, as shown in figure 3(d).

The CNN-AE is composed of a CNN encoder \mathcal{F}_e , which maps high-dimensional data into a low-dimensional space, and a CNN decoder \mathcal{F}_d , which extends the data low-dimensionalized by the encoder part. If a CNN-AE \mathcal{F}_c having a smaller latent vector $\tilde{\mathbf{q}}$ than the input \mathbf{q} can generate the output identical to the input, it means that the dimension can be successfully reduced while retaining the original information. Summarizing above, the procedures of the CNN-AE are expressed as

$$\mathbf{q}_{\text{deco}} \approx \mathcal{F}_c(\mathbf{q}), \quad \tilde{\mathbf{q}} = \mathcal{F}_e(\mathbf{q}), \quad \mathbf{q}_{\text{deco}} = \mathcal{F}_d(\tilde{\mathbf{q}}), \quad (7)$$

where \mathbf{q}_{deco} denotes the decoder output.

In the present study, a multi-scale CNN-AE model (MS-CNN-AE) shown in figure 4 is proposed to reduce the spatial dimension of flow field data. The MS-CNN-AE is inspired by the multi-scale CNN [30] developed for image-based super-resolution analysis to capture multi-scale sense of images. The size of three scales of filters are 3×3 , 5×5 , and 9×9 , respectively. As an example, the structure of the part to map the flow fields into the latent vector $\tilde{\mathbf{q}} \in \mathbb{R}^{6 \times 3 \times 4}$ (viz., the size of encoded values is $n_z = 72$) is summarized in table 1. There are batch normalization [31] layers between the convolution layer and the activation layer (ReLU) [32] to avoid the overfitting. The batch normalization, which normalizes the output of each unit based on the mean and variance in each training mini-batch, is known to accelerate learning by suppressing so-called internal covariate shift. The left and right parts of figure 4 are the encoder and the decoder, respectively. The flow fields fed as the input are mapped by these three scales of filters, and then three encoded values $\in \mathbb{R}^{6 \times 3 \times 4}$ are obtained. These three encoded values are added in the *add layer* shown in table 1, and fed into *7th Conv. layer* to obtain the encoded values representing the flow field in the low-dimensional space. Then, the decoder reconstructs the flow fields in the physical space from the encoded values using upsampling layers.

Usually, the objective of regression tasks with supervised machine learning is to obtain optimized weights \mathbf{W} by minimizing the predefined error function ε such that $\mathbf{W} = \text{argmin}_{\mathbf{W}} \|\varepsilon\|_{\gamma}$, where γ is the parameter of the norm. Here, we use a combination of the mean squared error ε_m and the gradient difference loss ε_g [33]

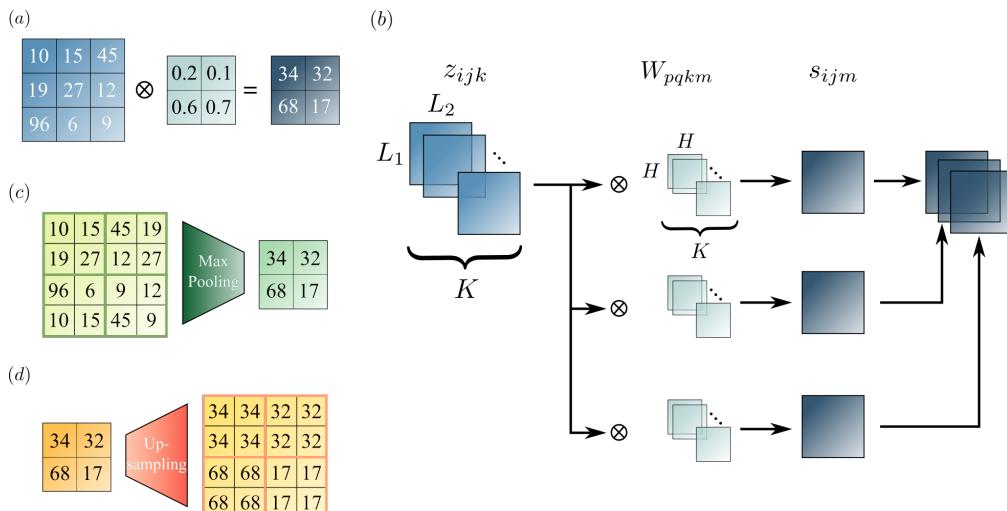


Fig. 3 Operations in the convolutional layer and the sampling layer: (a) convolutional operation using a weighted filter W ; (b) the computation in the convolution layer with $M = 3$; (c) max pooling operation. (d) upsampling operation.

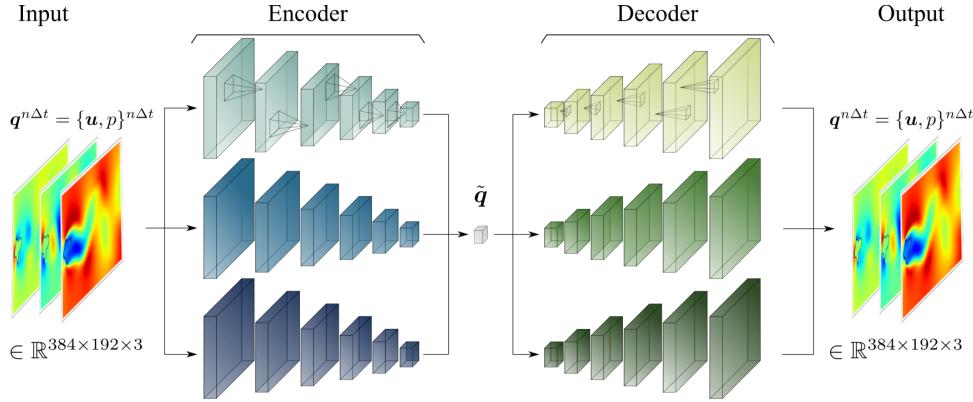


Fig. 4 Schematic of the MS-CNN-AE. The layers represented by cubes in the encoder part include the convolutional layer, the batch normalization layer, the ReLU layer, and the max pooling layer in order. As for the decoder part, the cubes include the convolutional layer, the batch normalization layer, the ReLU layer, and the upsampling layer in order.

Table 1 Structure of each CNN-AE.

Encoder		Decoder	
Layer	Output shape	Layer	Output shape
Input	(384, 192, 3)	8th Conv.	(6, 3, 4)
1st Conv.	(384, 192, 16)	Batch Normalization	(6, 3, 4)
Batch Normalization	(384, 192, 16)	ReLU	(6, 3, 4)
ReLU	(384, 192, 16)	Dispersion Layer	(6, 3, 4)
1st Max Pooling	(192, 96, 16)	1st Upsampling	(12, 6, 4)
2nd Conv.	(192, 96, 8)	9th Conv.	(12, 6, 4)
Batch Normalization	(192, 96, 8)	Batch Normalization	(12, 6, 4)
ReLU	(192, 96, 8)	ReLU	(12, 6, 4)
2nd Max Pooling	(96, 48, 8)	2nd Upsampling	(96, 48, 8)
3rd Conv.	(96, 48, 8)	10th Conv.	(24, 12, 8)
Batch Normalization	(96, 48, 8)	Batch Normalization	(24, 12, 8)
ReLU	(96, 48, 8)	ReLU	(24, 12, 8)
3rd Max Pooling	(48, 24, 8)	3rd Upsampling	(48, 24, 8)
4th Conv.	(48, 24, 8)	11th Conv.	(48, 24, 8)
Batch Normalization	(48, 24, 8)	Batch Normalization	(48, 24, 8)
ReLU	(48, 24, 8)	ReLU	(48, 24, 8)
4th Max Pooling	(24, 12, 8)	4th Upsampling	(96, 48, 8)
5th Conv.	(24, 12, 8)	12th Conv.	(96, 48, 8)
Batch Normalization	(24, 12, 8)	Batch Normalization	(96, 48, 8)
ReLU	(24, 12, 8)	ReLU	(96, 48, 8)
5th Max Pooling	(24, 12, 8)	5th Upsampling	(192, 96, 8)
6th Conv.	(12, 6, 4)	13th Conv.	(192, 96, 16)
Batch Normalization	(12, 6, 4)	Batch Normalization	(192, 96, 16)
ReLU	(12, 6, 4)	ReLU	(192, 96, 16)
6th Max Pooling	(6, 3, 4)	6th Upsampling	(384, 192, 16)
Add Layer	(6, 3, 4)	14th Conv.	(384, 192, 3)
7th Conv.	(6, 3, 4)	Batch Normalization	(384, 192, 3)
Batch Normalization	(6, 3, 4)	ReLU	(384, 192, 3)
ReLU	(6, 3, 4)	15th Conv. (Output)	(384, 192, 3)

as the loss function ε , i.e.,

$$\varepsilon = \varepsilon_m + \varepsilon_g, \quad (8)$$

$$\varepsilon_m = \frac{1}{\hat{N}_x} \frac{1}{\hat{N}_y} \frac{1}{N_\phi} \sum_{i=1}^{\hat{N}_x} \sum_{j=1}^{\hat{N}_y} \sum_{k=1}^{N_\phi} (q_{(i,j,k)} - q_{\text{deco}(i,j,k)})^2, \quad (9)$$

$$\begin{aligned} \varepsilon_g = & \frac{1}{\hat{N}_x} \frac{1}{\hat{N}_y} \frac{1}{N_\phi} \sum_{i=1}^{\hat{N}_x} \sum_{j=1}^{\hat{N}_y} \sum_{k=1}^{N_\phi} |(q_{(i,j,k)} - q_{(i-1,j,k)}) - (q_{\text{deco}(i,j,k)} - q_{\text{deco}(i-1,j,k)})| \\ & + |(q_{(i,j-1,k)} - q_{(i,j,k)}) - (q_{\text{deco}(i,j-1,k)} - q_{\text{deco}(i,j,k)})|, \end{aligned} \quad (10)$$

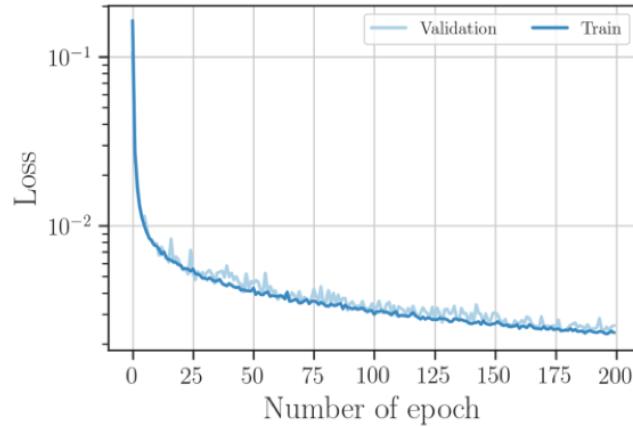


Fig. 5 An example of learning curve for the CNN-AE part.

where the subscripts represent the data indices. The gradient differential loss directly penalizes the gradient among grid points of the flow field data, and this feature enables the model to avoid blurry prediction [34]. Note that tuning of the weight between the mean squared error ε_m and the gradient differential loss ε_g is required, and its optimal weight varies depending on the problem. In this study, the weight is set to $\varepsilon_m : \varepsilon_g = 1 : 1$ following our preliminary test.

The Adam algorithm [35] is applied as the optimizer for weight updating, and a four-fold cross validation is applied to train the models and avoid overfitting [36]. The minibatch size is set at 100 — changing the minibatch size had no significant influence in our preliminary test. The number of epochs is fixed at 200 (i.e., no early stopping). Figure 5 shows an example of the learning curve, which presents the relation between the number of epochs and the loss value. The curve shows good convergence, and no overfitting is observed. In the model evaluation, we use the best model which provides the lowest validation loss.

2.2.2 Long Short-Term Memory (LSTM)

The long short-term memory (LSTM) [22] is a machine learning algorithm suited to handle time-series problems, e.g., speech recognition [38]. The LSTM layer is composed of a cell, an input gate, an output gate, and a forget gate, as illustrated in figure 6. The input gate is represented by d , output gate by o , and forget gate by g . The cell state is C and the cell output is given by h_t , while the cell input is denoted as x_t , where

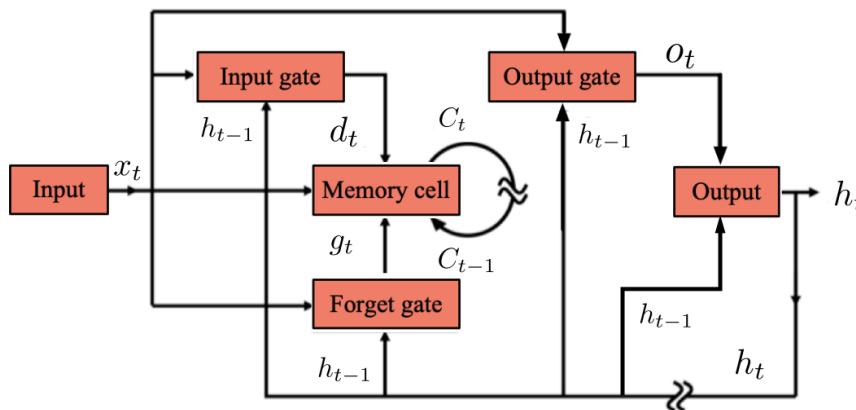


Fig. 6 Internal procedures of an LSTM.

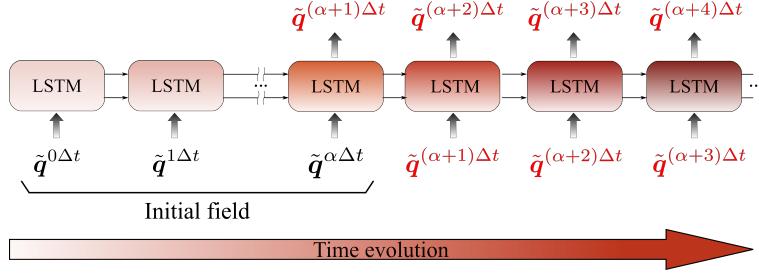


Fig. 7 Schematic of the prediction using the LSTM model. The encoded values in black letters are the initial fields generated by applying the CNN-AE to the DNS data. The encoded values indicate the predicted fields by the LSTM from the previous outputs of the LSTM or the initial fields. The number of the initial fields is $\alpha + 1$ in this figure.

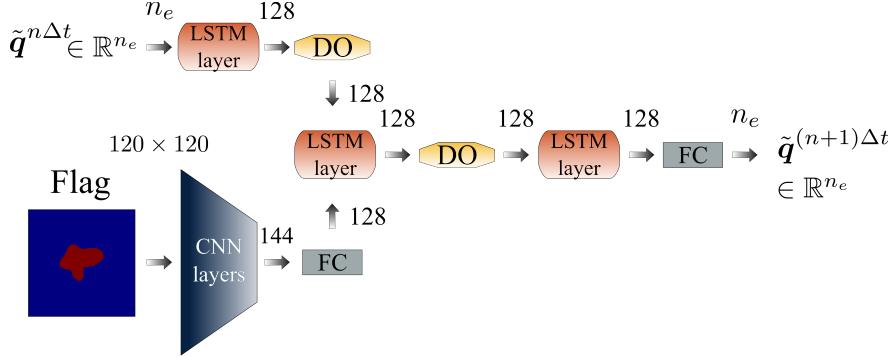


Fig. 8 Schematics of the LSTM model. DO and FC in this figure represent dropout layers and fully-connected layers, respectively. The values above the arrows indicate the number of input/output of those layers, and n_e represents the number of encoded values. Note that each LSTM layer has 128 units; viz., the output size of these layers is 128.

the subscripts represent a time step. In sum, the internal procedures of the LSTM are formulated as

$$d_t = \sigma(W_d \cdot [h_{t-1}, x_t] + \beta_d), \quad (11)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + \beta_o), \quad (12)$$

$$g_t = \sigma(W_g \cdot [h_{t-1}, x_t] + \beta_g), \quad (13)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + \beta_c), \quad (14)$$

$$C_t = g_t \times C_{t-1} + d_t \times \tilde{C}_t, \quad (15)$$

$$h_t = o_t \times \tanh(C_t), \quad (16)$$

where W represents the weights for each gates and β is the bias; the subscripts to C , e , and h represent the time indices, and σ is the sigmoid function. Although readers are referred to literature [22] for further details, this structure enables the LSTM layer to deal with the time-series problem by keeping the previous input information in the cell state.

In this study, an LSTM model is employed to predict the temporal evolution of low-dimensionalized flow fields generated by the CNN-AE as illustrated in figure 7. In the diagram, \tilde{q} denotes the low-dimensional field, and the superscript represents time indices. The arbitrary number of the flow fields are fed into the LSTM model as the initial encoded fields. Next, the field predicted from these initial fields is recursively incorporated as the input data to the LSTM model keeping the cell state. The details of the present LSTM model are summarized in figure 8. A dropout (DO) [39] is applied in order to avoid overfitting. A flag map of the bluff body (i.e., 1 for the bluff body region, 0 for the fluid region) is provided to the LSTM model as the information including the shape and boundary condition. Our preliminary test has shown that the model with the shape information outperforms the machine learning model without that information.

The mean squared error is used as the loss function $\tilde{\varepsilon}$ to train the LSTM model, i.e., $\tilde{\varepsilon} = \overline{(\tilde{\mathbf{q}}_{\text{true}} - \tilde{\mathbf{q}}_{\text{pred}})^2}$, where $\tilde{\mathbf{q}}_{\text{true}}$ is the true encoded field, $\tilde{\mathbf{q}}_{\text{pred}}$ is the field predicted by the LSTM model, and the overbar represents the average similar to equation (9). The solution data set is prepared from the output of the CNN-AE, and the LSTM model is trained using *teacher forcing* [40]. Following our preliminary test, the number of time sequences used for the training process is set to 20. Hence, the training for the LSTM model is equivalent to optimizing the weights in the LSTM model \mathbf{w}_L such that

$$\mathbf{w}_L = \operatorname{argmin}_{\mathbf{w}_L} \|\tilde{\mathbf{q}}^{(n+1)\Delta t} - \mathcal{F}_L(\tilde{\mathbf{q}}^{n\Delta t}, \tilde{\mathbf{q}}^{(n-1)\Delta t}, \tilde{\mathbf{q}}^{(n-2)\Delta t}, \dots, \tilde{\mathbf{q}}^{(n-19)\Delta t})\|_2, \quad (17)$$

where the subscript of “true” is omitted for brevity. Similarly to the CNN-AE above, the Adam algorithm [35] is applied as the optimizer, a four-fold cross validation is used, and the best model which provides the lowest validation loss in the learning process is used for the model evaluation. Both the minibatch size and the number of epochs are set to 100. An example of learning curve for the LSTM part is presented in figure 9, which shows good convergence and no overfitting.

For the model evaluation, the number of time steps used for the input to the LSTM \mathcal{F}_L is set to 1 such that $\tilde{\mathbf{q}}^{(n+1)\Delta t} = \mathcal{F}_L(\tilde{\mathbf{q}}^{n\Delta t})$ except for the first iteration. For the first iteration, the latent vector at the next time step is obtained from the solution data of the 5 initial time steps (i.e., $\alpha = 4$ in figure 7). In sum, the temporal evolution of the mapped vector in the LSTM is formulated as

$$\tilde{\mathbf{q}}^{5\Delta t} = \mathcal{F}_L(\tilde{\mathbf{q}}^{4\Delta t}, \tilde{\mathbf{q}}^{3\Delta t}, \tilde{\mathbf{q}}^{2\Delta t}, \tilde{\mathbf{q}}^{1\Delta t}, \tilde{\mathbf{q}}^{0\Delta t}), \quad (18)$$

$$\tilde{\mathbf{q}}^{(n+1)\Delta t} = \mathcal{F}_L(\tilde{\mathbf{q}}^{n\Delta t}), \quad n \geq 5. \quad (19)$$

Note that our preliminary test has shown that the results are not sensitive to the number of time steps used for the first iteration.

2.2.3 Machine-learning based reduced order model (ML-ROM)

As illustrated in figure 10, the proposed machine-learning based reduced order model (ML-ROM) is a combination of the MS-CNN-AE model and the LSTM model introduced above. The initial flow fields generated by DNS are fed into the trained CNN encoder to map those into the latent space. By feeding the obtained latent vectors to the trained LSTM model, it predicts the latent vector at the next time step. The LSTM model recursively predicts the temporal evolution of the encoded fields by using the previous output as the input. The temporal evolution of the flow field in the physical space can be recovered by using the trained CNN decoder. Note that the number of initial flow fields in this figure is set to 1 for simplicity of illustration.

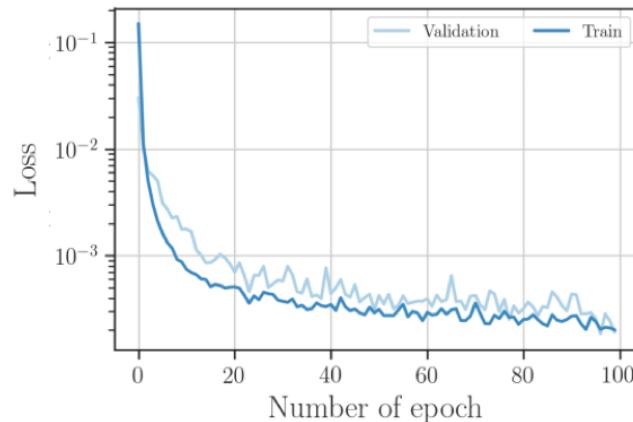


Fig. 9 An example of learning curve for the LSTM part.

3 Results and Discussion

3.1 Assessment of ML-ROM for wakes behind various random shapes

As a proof of concept to establish an ML-ROM for unseen data, we use the data sets of bluff bodies with various random shapes, as explained in section 2. In this subsection, the MS-CNN-AE is developed first to map the high dimensional flow $\mathbb{R}^{384 \times 192 \times 3}$ into a latent space $\mathbb{R}^{6 \times 3 \times 4}$. Then, the LSTM part is trained to learn the temporal evolution of the obtained latent vectors. Note that the dependence on the latent vector size will be examined in the next subsection.

The MS-CNN-AE is trained by using the data set which consists of flow data for 80 different bluff bodies with the 500 instantaneous time-series fields prepared for each bluff body shape. This model is evaluated by the test data set, which are different from those used for training. The test data set includes flows around bluff bodies for 20 different shapes shown in figure 11.

The flow fields computed by the DNS and those reconstructed by the MS-CNN-AE are summarized in figure 12. In this figure, the flows with shape numbers of 1, 3, 5, 7, 9, 11, 13 and 15 are shown as the examples. The reconstructed flow fields show good agreement with the reference DNS fields. Although not shown here, the results with other bluff body shapes have similar trends to figure 12. Time-averaged local squared error fields for shape number 1 are shown in figure 13. Although the error is concentrated near the bluff body, the error is sufficiently small in the wake region.

The mean streamwise velocities on the centerline of the wake are presented in figure 14(a). The reconstructed centerline velocities are in excellent agreement with the reference DNS data. The mean squared errors, the time-averaged drag and lift coefficients are also summarized in figures 14(b), (c) and (d), which

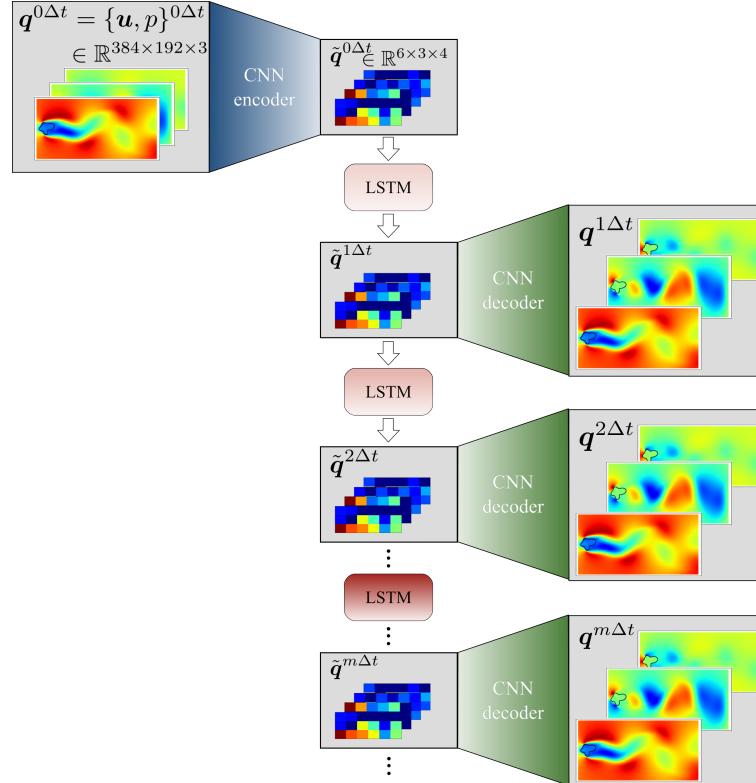


Fig. 10 Schematic of the ML-ROM with the latent space size of $6 \times 3 \times 4$. The number of time steps of the initial field is set to 1 for illustration purpose. The compressed vector obtained by using the CNN encoder evolves temporally using LSTM. The temporal evolution of the flow field is recovered by using the CNN decoder.

indicate that the mean squared errors are sufficiently small and the averaged force coefficients of the reconstructed fields reasonably match the DNS values.

The LSTM is trained by using the time step of $\Delta t = 0.25$ to learn the temporal evolution of the low-dimensionalized fields for the 80 different bluff bodies obtained by the MS-CNN-AE to construct the ML-ROM, as illustrated in figure 10. The amount of the training and validation data is 40000, which consist of 500 time-series data for each bluff body. Five instantaneous flows are prepared for each shape as the initial fields of the predictions, as mentioned above. Some instantaneous fields predicted by the ML-ROM after 100 recursive iterations corresponding to $t = 25$ are compared to the DNS data in figure 15. Both flows are observed to be similar for all attributes.

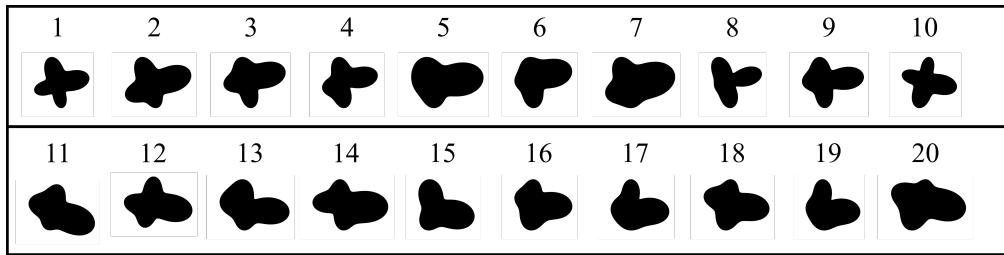


Fig. 11 The bluff body shapes of the test data set used to evaluate the machine learning models. The number shown above each shape represents the shape number.

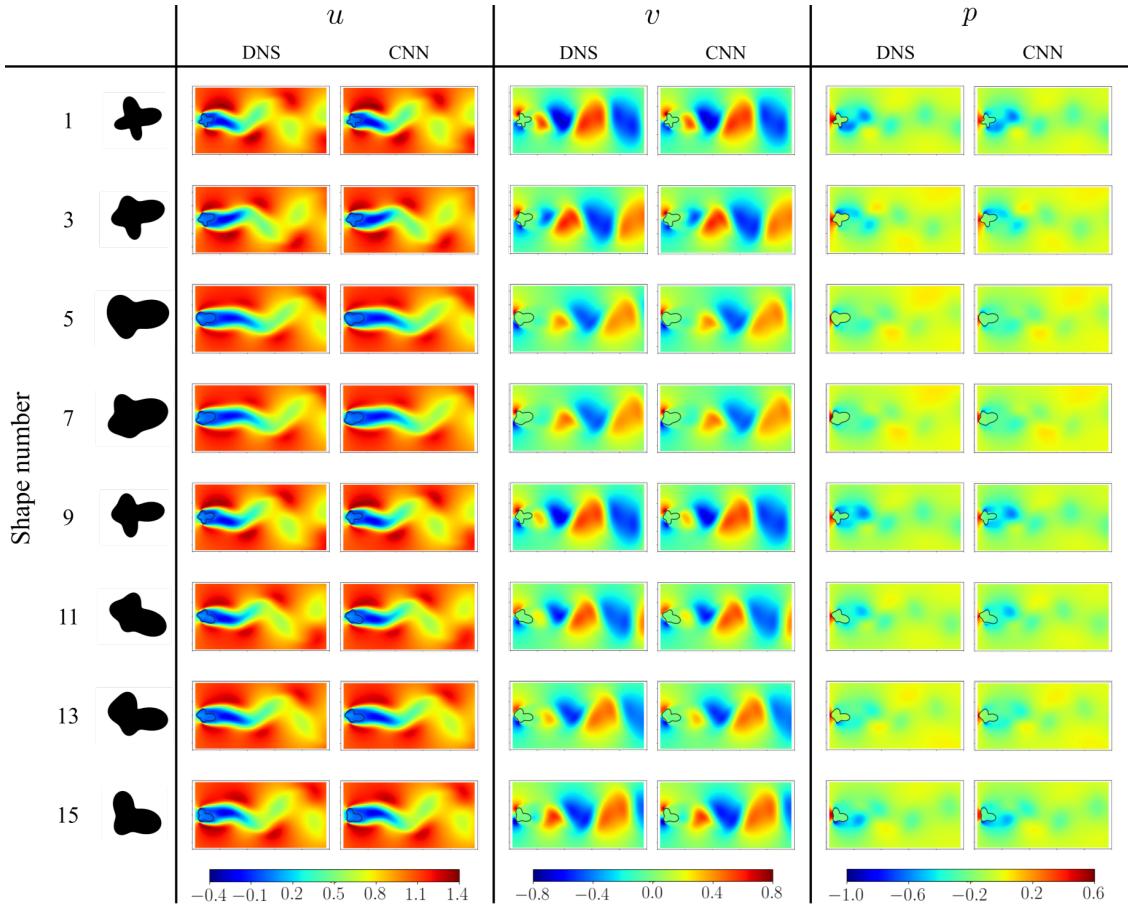


Fig. 12 Instantaneous flow fields for various bluff bodies. Flow fields computed by the DNS and those reconstructed by the MS-CNN-AE model are compared.

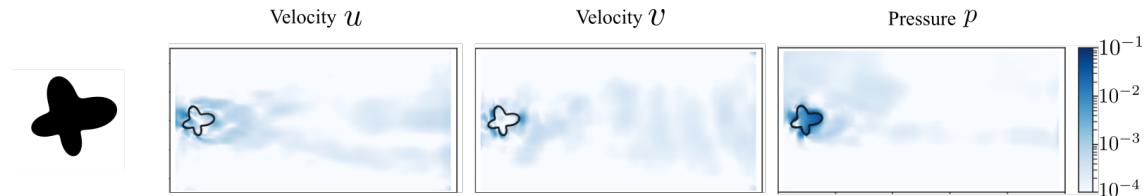


Fig. 13 Time-averaged local squared error fields of MS-CNN-AE for shape number 1.

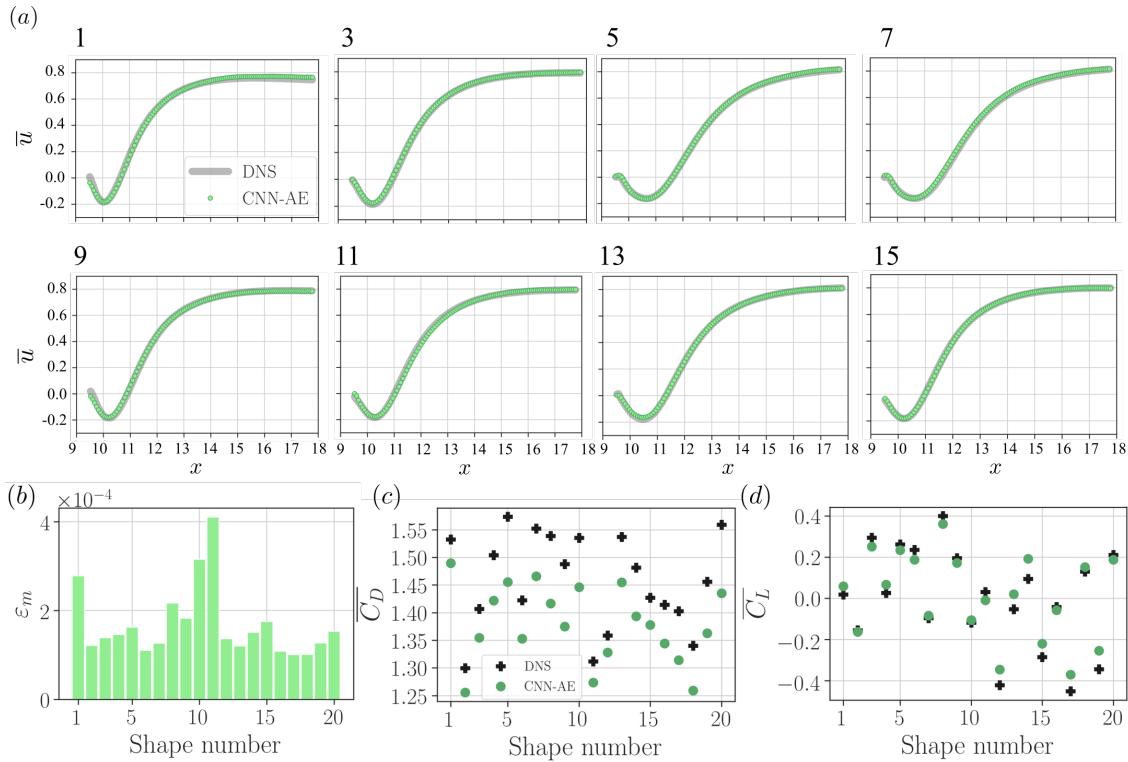


Fig. 14 Assessments of the MS-CNN-AE model for flows around various bluff bodies at $Re_D = 100$: (a) mean streamwise velocities on the centerline for shape numbers 1, 3, 5, 7, 9, 11, 13 and 15; (b) mean squared errors against the reference DNS data; (c) time-averaged drag coefficient; (d) time-averaged lift coefficient.

The statistical assessments of the prediction by the ML-ROM are summarized in figure 16. The predicted results are again in good agreement with the reference DNS data in terms of the mean centerline velocity and the force coefficients, which suggests that the present ML-ROM can successfully capture the feature of the unsteady wake. As shown in figure 16(d), the Strouhal number St is also well predicted, which confirms that the temporal structure is also well reproduced by the LSTM part even for the flows not used for the training (note again that shapes 1–20 are not used in the training process).

We also present in figure 17 the time-averaged local squared error computed using 1000 recursive inputs. Because of the recursive input, the time-averaged error is concentrated in the wake region, especially where the fluctuations are large. The time trace of the mean squared error is also shown in figure 18. The error varies periodically in time due to the small difference in the Strouhal number (figure 16(d)), but it does not grow. Summarizing above, the present ML-ROM is confirmed to have the ability to predict the flows around various bluff bodies.

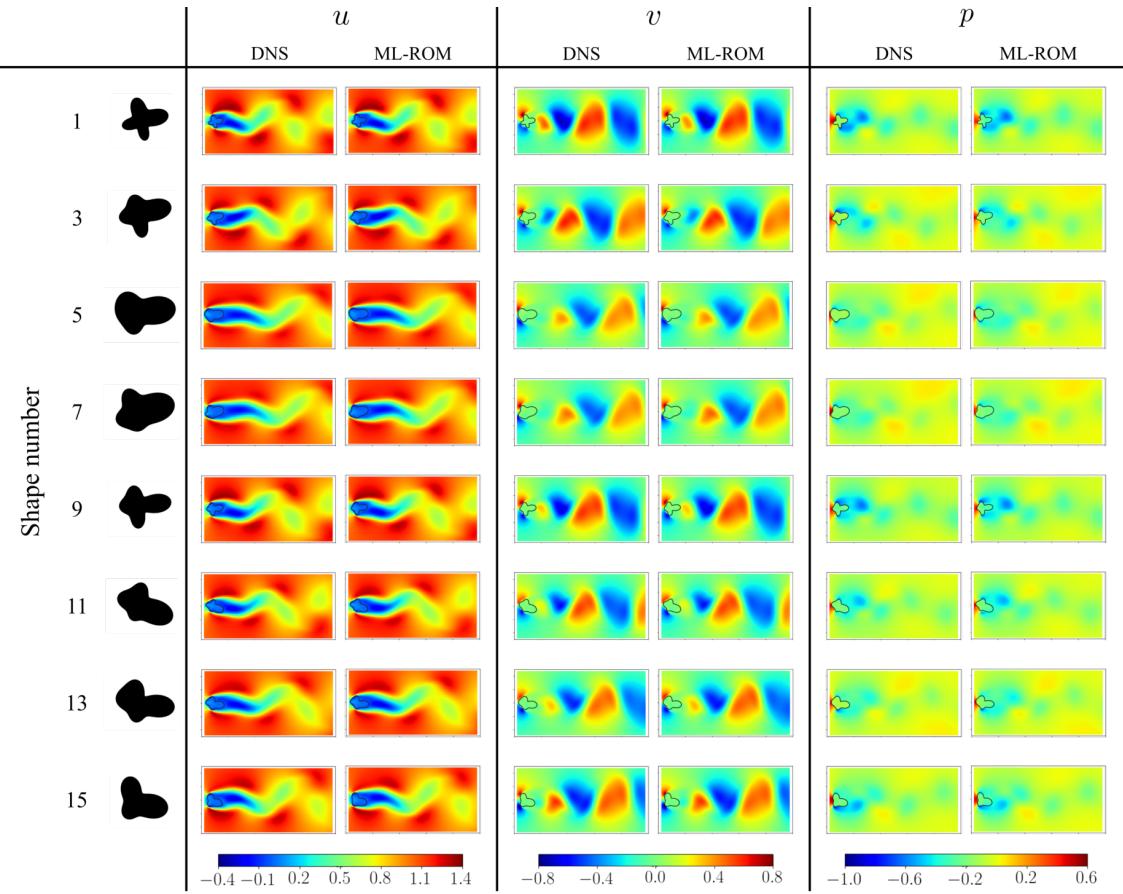


Fig. 15 Instantaneous flow fields with various bluff bodies at $t = 25$. The DNS and reconstructed flow fields by the ML-ROM are summarized.

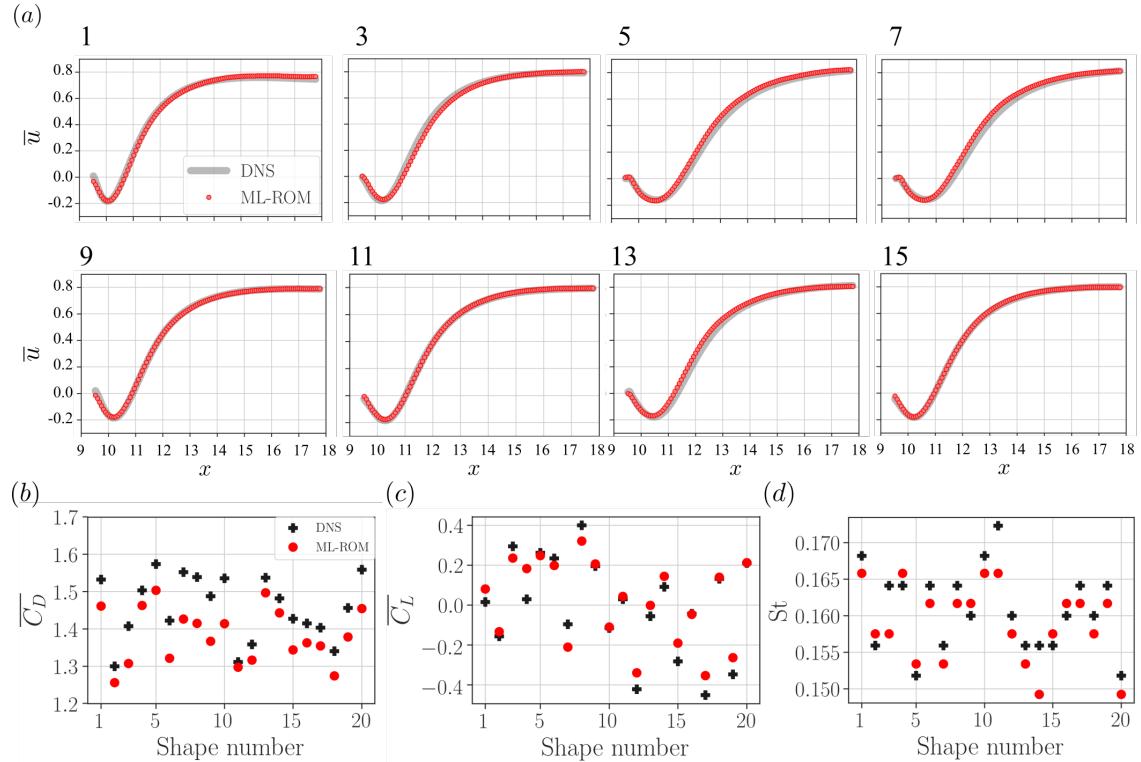


Fig. 16 Assessments of ML-ROM with various shapes wake at $Re_D = 100$: (a) mean streamwise velocity on the centerline for shape numbers 1, 3, 5, 7, 9, 11, 13 and 15; (b) drag coefficient; (c) lift coefficient; (d) Strouhal number.

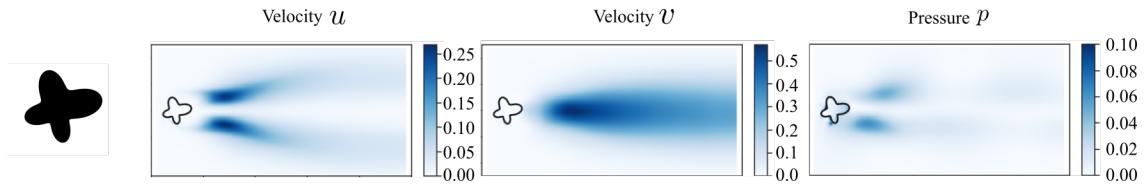


Fig. 17 Time-averaged local squared error fields of ML-ROM for shape number 1.

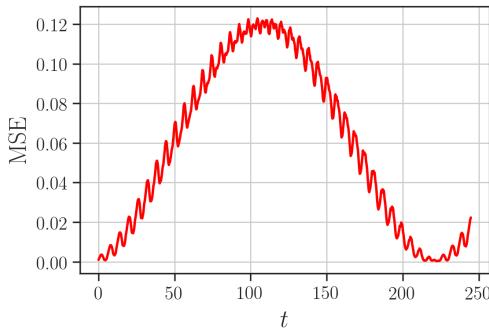


Fig. 18 Time trace of mean squared error of ML-ROM for shape number 1.

3.2 Influence of the parameters

In the aforementioned discussion, we have set the size of the latent vector in the MS-CNN-AE to be $n_z = 72$ ($= 6 \times 3 \times 4$) and the time steps in between the mapped vectors for the LSTM to be $\Delta t = 0.25$. In this subsection, we discuss the influence of these parameters.

3.2.1 Dependence on the latent vector size in the MS-CNN-AE

The dependence on the latent vector size n_z in the MS-CNN-AE is investigated and summarized in figure 19. Here, we examine $n_z = 2, 36, 72$ (baseline), and 4608. Since the temporal evolution of the mapped vector is obtained by the LSTM, which has a fully-connected structure between layers, a smaller latent vector allows us to establish an ML-ROM at a lower computational cost.

As shown in figure 19(a), the mean centerline velocity looks reasonably well reproduced in all cases. However, the mean velocities for some shapes, i.e., shapes 1, 5, and 7, are underestimated with $n_z = 2$ and 36. Similar trends can also be seen in the assessment of the force coefficients as summarized in figures 19(c) and (d). It suggests that $n_z = 72$ is the minimum size required to reconstruct the present flow fields with an appropriate fidelity. It is also surprising that the error ε with $n_z = 72$ is smaller than that with $n_z = 4608$, as shown in figure 19(b). This is likely due to the structure of the CNN-AE, which has more pooling operations for $n_z = 72$ case. It is widely known that incorporating the pooling operation in CNN structures enables the models to retain the robustness against the rotation or translation of the images because the sensitivity is decreased [16]. It indicates that the model with $n_z = 72$ is better than that with $n_z = 4608$ in terms of generality for unknown wakes thanks to the aforementioned robustness, especially in the present case where the wakes behind random shapes are considered.

Summarizing above, over-compression of the input and output flow data has a risk to lack the spatially coherent information of the flow field because of the pooling operation; however, the appropriate number of the pooling operation allows us to keep the robustness for unseen data.

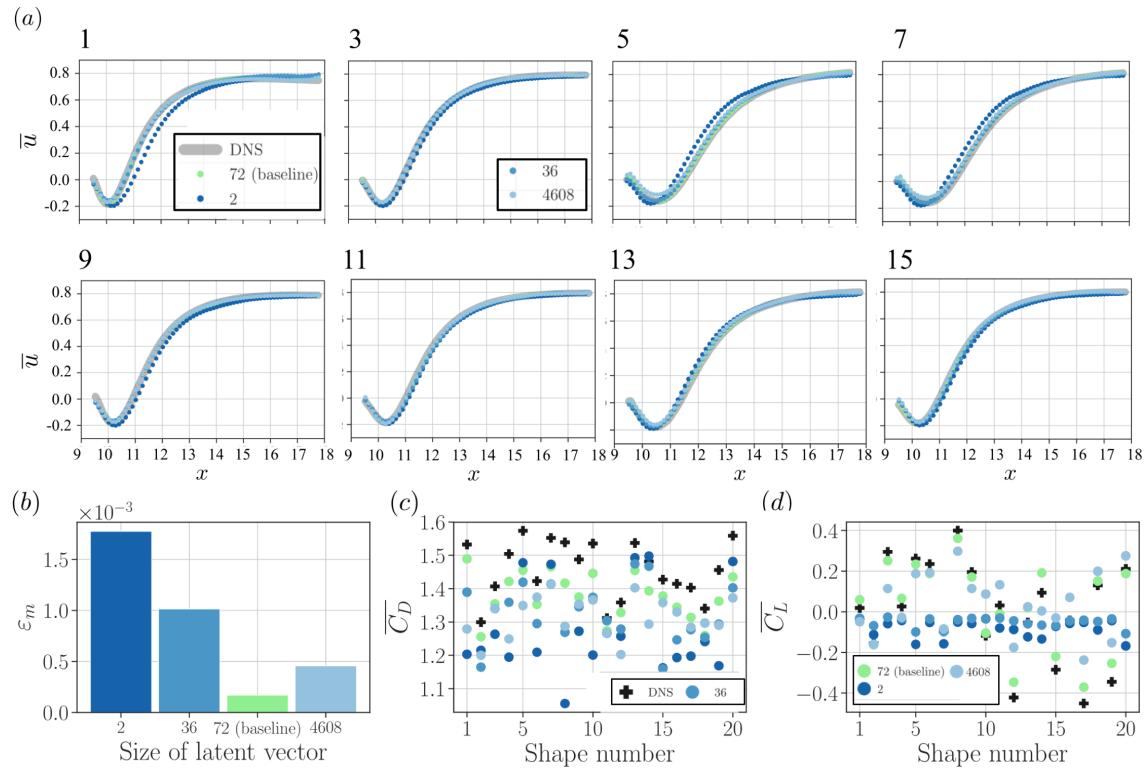


Fig. 19 Dependence on the latent vector size in the MS-CNN-AE: (a) mean streamwise velocity on the centerline; (b) mean squared error; (c) time-averaged drag coefficient; (d) time-averaged lift coefficient. Here, four-fold cross validation are arranged.

3.2.2 Dependence on the time step size in the LSTM

For high-fidelity simulations such as DNS and large eddy simulation, the time step size is always limited by numerical constraints. Thus, it would be attractive if the present ML-ROM can be used with substantially wider time step sizes.

Let us examine the dependence on the time step size in the LSTM, as summarized in figure 20. Here, we consider 11 cases: $\Delta t = 0.25$ (baseline)–5.25 with an increment of 0.50 in dimensionless time, although only the cases with $\Delta t = 0.25, 1.25, 2.75, 3.75$, and 5.25 are shown in figures 20(a), (c), (d), and (e). Recall that the time step used in the DNS was $\Delta t = 2.5 \times 10^{-3}$; namely, the baseline time step of $\Delta t = 0.25$ used for the LSTM is already 100 times wider than that. As shown here, the basic trend observed for all assessments is that the error increases with the time step size, especially for $\Delta t = 2.75$ and 3.75.

It is worth noting that the mean centerline velocity profile and the force coefficients are in reasonable agreement even with $\Delta t = 5.25$. However, in this case, the ML-ROM is considered to learn a typical aliasing signal because the sampling interval $\Delta t = 5.25$ is close to one period of the actual flow $T \simeq 6$. The Strouhal number predicted with $\Delta t = 5.25$ are around $St \simeq 0.02$ for all Reynolds numbers as shown in figure 20(e), which is also consistent with the value for the -1 aliasing at this sampling rate (i.e. $|1/T - 1/\Delta t| \simeq 0.02$). A similar argument holds for the cases of $\Delta t = 3.75$ or 2.75 where the sampling interval is longer than or just around the interval corresponding to the Nyquist frequency of the present periodic signal.

We note in passing that the results of the present ML-ROM also depends on the number of time steps used for the input of LSTM to predict the field of the next time step. We used 20 time steps for the training process of LSTM, but significant dependence was not observed in our preliminary test as far as more than 5 time steps were used. This is likely due to the periodic nature of this specific flow. Otherwise, the number of

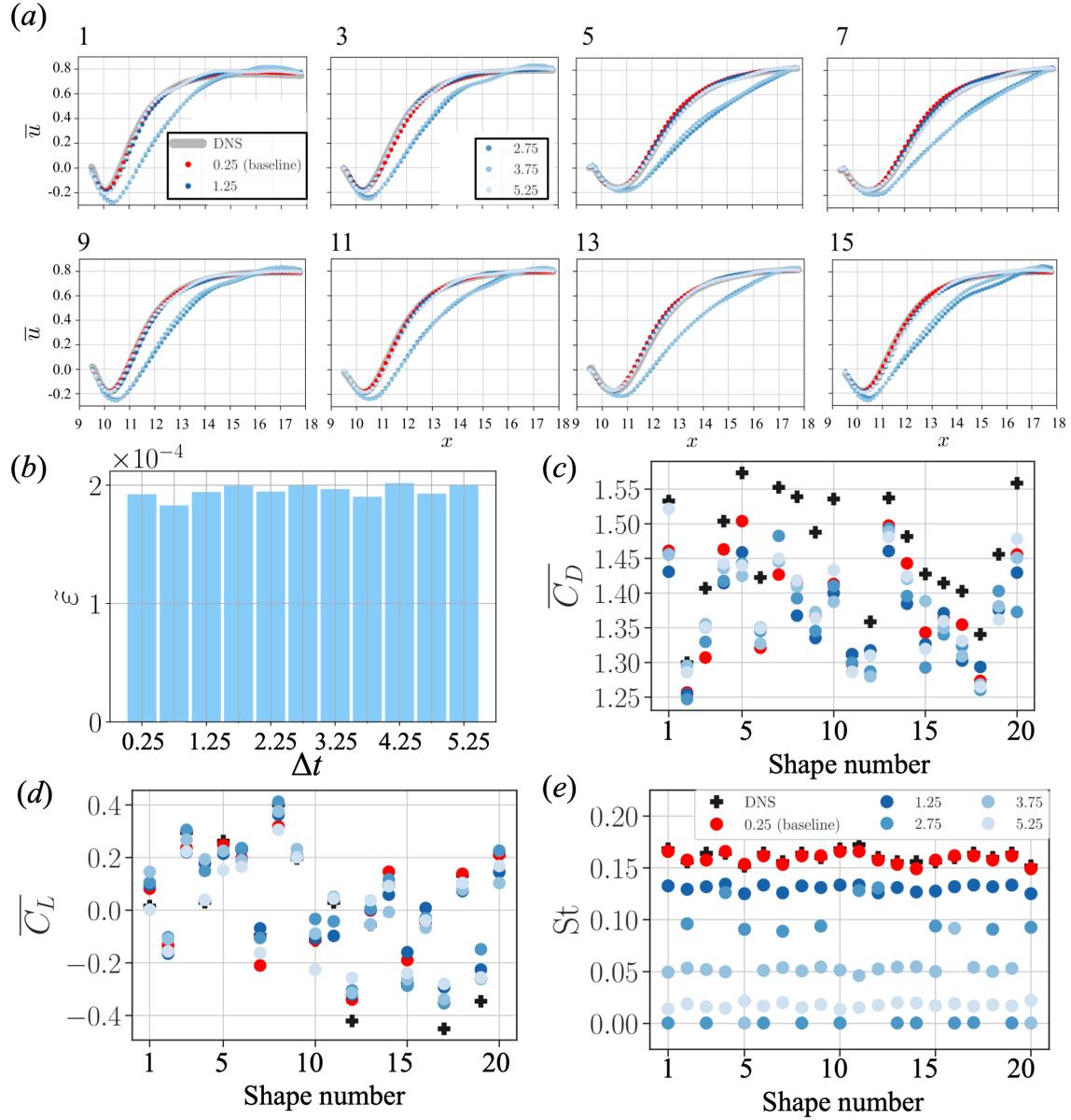


Fig. 20 Dependence on the time step size in the LSTM: (a) mean streamwise velocity on the centerline; (b) relationship between time step and the L_2 error; (c) time-averaged drag coefficient; (d) time-averaged lift coefficient; (e) Strouhal number. Here, four-fold cross validation are arranged.

input time steps used for the training process is also a crucial factor, and it should be chosen depending on user's requirements.

4 Conclusions

We presented machine-learning-based reduced order modeling for unsteady flows. A convolutional neural network based autoencoder (CNN-AE) was employed to map a high-dimensional flow field into a low-dimensional latent space, and a long short term memory (LSTM) was utilized to deal with the temporal evolution of the low-dimensionalized vectors obtained by the CNN-AE. As a test case, flows around bluff bodies with various

shapes were considered. The flows predicted by the machine-learned reduced order model (ML-ROM) showed statistically good agreement with the reference DNS data also for unseen bluff body shapes not used in the training process, which suggests that the present ML-ROM learns not just the flow fields used for training but the physics governed by the Navier–Stokes equation under different geometrical configurations.

Moreover, some case studies were conducted to investigate the dependence on the parameters used for the ML-ROM. The size of the latent vector of the CNN-AE model has relatively small influence on the reconstruction ability, but this might be specific to the present problem with temporal periodicity. We also found that the structure of the CNN-AE allows us to keep the robustness for unseen flow data. Concerning the dependence on the time step size used in the LSTM, the error increased with the time step size between the mapped vectors. The value of $\Delta t = 0.25$, which corresponds to about 20 subdivision of one period of vortex shedding, can be recommended from the present study to reproduce the Strouhal number accurately.

The present study was a proof of concept to establish an ML-ROM for more general fluid dynamics. It should be stressed again, however, that the present proof of concept was performed with a limited range of shapes, and that more variability will be required in practice. Although laminar periodic flows are considered as the problem setting in the present study, the proposed idea can be further extended to more complex phenomena, e.g., three dimensional flows at high Reynolds numbers. Concerning the possibility of applying LSTM to turbulent flows, Srinivasan et al. [20] have recently demonstrated that the chaotic temporal evolution of the nine-equation turbulent shear flow model can be well captured by utilizing the LSTM, as mentioned in the introduction. Therefore, the key issue for the present type of ML-ROM to be applied to more complex flows should be the development of a more efficient — and preferably interpretable — low-dimensionalization method, as is tackled by different research groups [21, 41].

Acknowledgements Authors are grateful to Dr. S. Obi, Dr. K. Ando, Mr. M. Morimoto and Mr. T. Nakamura (Keio University) for fruitful discussions. This work was supported through JSPS KAKENHI Grant Number 18H03758 by Japan Society for the Promotion of Science.

References

1. K. Taira, S. L. Brunton, S. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis and L. S. Ukeiley, Modal analysis of fluid flows: An overview, *AIAA J.*, 55, 12, 4013–4041 (2017).
2. J. L. Lumley, The structure of inhomogeneous turbulent flows, In *Atmospheric turbulence and wave propagation*, eds. Yaglom, A. M. and Tatarski, V. I., Moscow, Nauka, 166–178 (1967).
3. P. Schmid, Dynamic mode decomposition of numerical and experimental data, *J. Fluid Mech.*, 656, 5–28 (2010).
4. H. P. Bakewell and J. L. Lumley, Viscous sublayer and adjacent wall region in turbulent pipe flow, *Phys. Fluid*, 10, 1880 (1967).
5. F. Gómez and H. M. Blackburn, Data-driven approach to design of passive flow control strategies, *Phys. Rev. Fluids*, 2, 021901 (2017).
6. G. Alfonsi and L. Primavera, The structure of turbulent boundary layers in the wall region of plane channel flow, *Proc. R. Soc. A*, 463, 2078, 593–612 (2007).
7. G. E. Hinton and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Nature*, 313, 504–507 (2006).
8. Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature*, 521, 436–444 (2015).
9. N. Kutz, Deep learning in fluid dynamics, *J. Fluid Mech.*, 814, 1–4 (2017).
10. S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, *Annu. Rev. Fluid Mech.*, 52, 477–508 (2019).
11. K. Taira, M. S. Hemati, S. L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S. T. M. Dawson, and C. A. Yeh, Modal analysis of fluid flows: Applications and outlook, *AIAA J.*, <https://doi.org/10.2514/1.J058462> (2019).
12. M. P. Brenner, J. D. Eldredge, J. B. Freund, Perspective on machine learning for advancing fluid mechanics, *Phys. Rev. Fluids*, 4, 100501 (2019).
13. K. Fukami, K. Fukagata, and K. Taira, Assessment of supervised machine learning methods for fluid flows, *Theor. Comput. Fluid Dyn.* (2020). <https://doi.org/10.1007/s00162-020-00518-y>
14. 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).
15. K. Fukami, K. Fukagata, and K. Taira, Super-resolution reconstruction of turbulent flows with machine learning, *J. Fluid Mech.* 870, 106–120 (2019).
16. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE*, 86(11), 2278–2324 (1998).

17. J. Xiaowei, C. Peng, C. Wen-Li, and L. Hui, 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, 047105 (2018).
18. J. Viquerata., E. Hachema, A supervised neural network for drag prediction of arbitrary 2D shapes in low Reynolds number flows, arXiv:1907.05090 (2019).
19. O. San and R. Maulik, Extreme learning machine for reduced order modeling of turbulent geophysical flows, *Phys. Rev. E* 97, 04322 (2018).
20. P. A. Srinivasan, L. Guastoni, H. Azizpour, P. Schlatter, and R. Vinuesa, Predictions of turbulent shear flows using deep neural networks. *Phys. Rev. Fluids*, 4, 054603 (2019).
21. T. Murata, K. Fukami, and K. Fukagata, Nonlinear mode decomposition with convolutional neural networks for fluid dynamics, *J. Fluid Mech.* 882, A13 (2020).
22. S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, 9, 1735–1780 (1997).
23. Y. Anzai, K. Fukagata, P. Meliga, E. Boujo, and F. Gallaire, Numerical simulation and sensitivity analysis of a low-Reynolds-number flow around a square cylinder controlled using plasma actuators, *Phys. Rev. Fluids*, 2, 043901 (2017).
24. H. Kor, M. B. Ghomizad, and Fukagata, K, A unified interpolation stencil for ghost-cell immersed boundary method for flow around complex geometries, *J. Fluid Sci. Technol.*, 12, JFST011 (2017).
25. A. A. Amsden and F. H. Harlow, The SMAC method, Los Alamos Scientific Lab. Rep., No. LA-4370 (1970).
26. P. R. Spalart, R. D. Moser, and M. M. Rogers, Spectral methods for the Navier-Stokes equations with one infinite and two periodic directions, *J. Comput. Phys.*, 96, 297–324 (1991).
27. A. Mitsuishi, K. Fukagata, and N. Kasagi, Near-field development of large-scale vortical structures in a controlled confined coaxial jet, *J. Turbul.*, 8, N23, 1–27 (2007).
28. K. Fukami, Y. Nabaue, K. Kawai, K. Fukagata, Synthetic turbulent inflow generator using machine learning, *Phys. Rev. Fluids*, 4, 064603 (2019).
29. G. E. Hinton, and R. R. Salakhutdinov., Reducing the dimensionality of data with neural networks, *Science*, 313, 504–507 (2006).
30. X. Du, X. Qu, Y. He, and D. Guo, Single image super-resolution based on multi-scale competitive convolutional neural network, *Sensors*, 18, No. 789, 1-17 (2018).
31. S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, the 32 nd International Conference on Machine Learning, Lille, France (2015).
32. V. Nair and G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, In Proc. 27th International Conference on Machine Learning (2010).
33. M. Mathieu, C. Couprie, and Y. LeCun, Deep multi-scale video prediction beyond mean square error, arXiv preprint, arXiv:1511.05440v6 [cs.LG] (2016).
34. 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).
35. D. Kingma, and J. Ba, A method for stochastic optimization, arXiv preprint, arXiv:1412.6980 [cs.LG] (2014).
36. S. L. Brunton and J. N. Kutz, Data-driven science and engineering: Machine learning, dynamical systems, and control, Cambridge Univ. Press (2019).
37. L. Prechelt, Automatic early stopping using cross validation: quantifying the criteria, *Neural Networks*, 11, 4, 761–767 (1998).
38. A. Graves, N. Jaitly, and A. Mohamed, Hybrid speech recognition with deep bidirectional LSTM, 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech, Dec. 8-12 (2013).
39. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.*, 15, 1929–1958 (2014).
40. R. J. Williams and D. Zipser, A learning algorithm for continually running fully recurrent neural networks, *Neural Comput.*, 1, 270–280 (1989).
41. A. Ehlert, C. N. Nayeri, M. Morzynski, B. R. Noack, Locally linear embedding for transient cylinder wakes, arXiv preprint, arXiv:1906.07822 [physics.flu-dyn] (2019).