Theor. Comput. Fluid Dyn. (2021) 35:633–658
https://doi.org/10.1007/s00162-021-00580-0

ORIGINAL ARTICLE

Masaki Morimoto · Kai Fukami · Kai Zhang · Aditya G. Nair · Koji Fukagata

Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization

Received: 8 February 2021 / Accepted: 21 July 2021 / Published online: 2 August 2021

Abstract We focus on a convolutional neural network (CNN), which has recently been utilized for fluid flow analyses, from the perspective on the influence of various operations inside it by considering some canonical regression problems with fluid flow data. We consider two types of CNN-based fluid flow analyses: (1) CNN metamodeling and (2) CNN autoencoder. For the first type of CNN with additional scalar inputs, which is one of the common forms of CNN for fluid flow analysis, we investigate the influence of input placements in the CNN training pipeline. As an example, estimation of drag and lift coefficients of an inclined flat plate and two side-by-side cylinders in laminar flows is considered. For the example of flat plate wake, we use the chord Reynolds number Re_c and the angle of attack α as the additional scalar inputs to provide the information on the complexity of wake. For the wake interaction problem comprising flows over two side-by-side cylinders, the gap ratio and the diameter ratio are utilized as the additional inputs. We find that care should be taken for the placement of additional scalar inputs depending on the problem setting and the complexity of flows that users handle. We then discuss the influence of various parameters and operations on the CNN performance, with the utilization of autoencoder (AE). A two-dimensional decaying homogeneous isotropic turbulence is considered for the demonstration of AE. The results obtained through the AE highly rely on the decaying nature. Investigation on the influence of padding operation at a convolutional layer is also performed. The zero padding shows reasonable ability compared to other methods which account for the boundary conditions assumed in the numerical data. Moreover, the effect of the dimensional reduction/extension methods inside CNN is also examined. The CNN model is robust against the difference in dimension reduction operations, while it is sensitive to the dimensional extension methods. The findings of this paper will help us better design a CNN architecture for practical fluid flow analysis.

Keywords Convolutional neural network · Machine learning · Fluid flows · Metamodeling · Autoencoder

Communicated by Luca Magri.

M. Morimoto (✉) · K. Fukagata · K. Fukami
Department of Mechanical Engineering, Keio University, Yokohama 223-8522, Japan
E-mail: masaki.morimoto@kflab.jp

K. Fukami
Department of Mechanical and Aerospace Engineering, University of California, Los Angeles, CA 90095, USA

K. Zhang
Department of Mechanical and Aerospace Engineering, Rutgers University, Piscataway, NJ 08854, USA

A. G. Nair
Department of Mechanical Engineering, University of Nevada, Reno, NV 89557, USA

1 Introduction

Convolutional neural networks (CNNs) have recently been recognized as a powerful tool to analyze complex fluid flow phenomena [1]. Although they are still in the phase of fundamental studies, the great potential of CNN-based analyses can be found in turbulence modeling [2], data reconstruction [3,4], and low dimensionalization [5]. However, current uses of CNNs require not only the tuning of a bunch of parameters but also the incorporation of *a priori* knowledge into its modeling. Due to these issues, the recent uses of CNNs for fluid flows are based on trial-and-error iterations by users because of a lack of guidelines. In this paper, we address the aforementioned issues by focusing on a general form of CNN-based regression tasks for fluid flow analysis $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \phi)$, where $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, indicate input and output data of machine learning model $\mathcal{F}$ with additional scalar inputs ϕ.

One of the most active areas for applying CNN for fluid flow analyses is turbulence modeling [2,6]. Lapeyre et al. [7] utilized a U-net-based CNN for estimating sub-grid scale reaction rates in their large eddy simulation (LES) framework. At that moment, previous studies regarding NNs and LES sub-grid modeling had been conducted using a multilayer perceptron (MLP) [8]. Hence, the use of CNN for the LES closure was one of the novelties of their study. The comparison between the MLP and the CNN for LES closure modeling is well discussed by Pawar et al. [9]. The CNN can also be applied to turbulence modeling for Reynolds-averaged Navier–Stokes (RANS) simulations. Thuerey et al. [10] considered CNN-based estimation for the airfoil flow field obtained from RANS and showed that NNs are even applicable to RANS data, dispelling the skepticism on the applications of NNs to RANS. In addition to the aforementioned efforts on LES and RANS, Font et al. [11] have recently proposed new concept for the decomposition of Navier–Stokes equation called *spanwise-averaged N–S equation*. Analogous to LES and RANS equations, it also has a closure term. They used a CNN-based regressor to close that term with a velocity field while considering the example of wakes behind a cylinder and an ellipse.

CNN-based fluid flow data reconstruction can also be regarded as a promising field. For instance, the CNN-based super-resolution analysis proposed by Fukami et al. [12] is a proof of concept of fluid flow data recovery with machine learning. The aim of the super-resolution analysis is to reconstruct high-resolution data from its low-resolution counterpart. Since this low-resolution part can also be replaced by various forms of low-resolution fluid flow information, e.g., local sensor measurements and limited availability of data, the super-resolution idea has been extended to not only numerical [13,14] but also experimental studies [15–17]. From the view of estimation, the combination of CNN and MLP can widely be seen for their purposes, since their output shape is often in a form of scalar. For example, Salehipour and Peltier [18] attempted to predict the small-scale structures in the ocean turbulence called *atoms* using a CNN. Otherwise, Morimoto et al. [19] visualized the internal procedures of the CNN-MLP model with fluid flow analyses toward practical applications from the perspective on interpretability.

Furthermore, the CNN-based modeling is also capable of low-dimensionalizing fluid flows, as a form of autoencoder (AE) [20]. The AE has the same output data as the input while having a dimension compression procedure inside its structure. Because the AE is trained to output the same data as the input, the latent variables, which can be extracted from the bottleneck layer inside the AE, can be regarded as a low-dimensionalized representation of high-dimensional data, if the reconstruction via the AE is well performed. To the best of our knowledge, Milano and Koumoutsakos [21] first brought the idea of AE into the fluid dynamics field. Although their model was based on the MLP, the CNN has also recently been applied to build AE thanks to the CNN's great advantage against the MLP in terms of the number of weights inside the model while being able to keep its accuracy [22,23]. In addition, the extracted low-dimensional representations via AE can also be utilized for the construction of reduced-order modeling which has a similar form to that of the proper orthogonal decomposition-based Galerkin integration [24–28].

As discussed above, we are now able to appreciate the strong potential and applicability of CNN for fluid flow analyses. However, the current success of CNN and fluid flow analysis is based on trial-and-error iterations by fluid mechanicians since we have no guideline for parameter decisions inside the CNN. For instance, the work on super-resolution analysis introduced above [12] reported the importance of the utilization of multi-size filters inside CNNs so as to account for a wide range of scales included in turbulence. A similar idea can also be found in the construction of NN-based reduced order modeling [29,30] and surrogate models for high-fidelity simulations [31]. Otherwise, several reports utilize additional scalar inputs which highly relates to fluid flow phenomena, e.g., angle of attack, Reynolds number, and bluff body shapes, to improve the estimation or low-dimensionalization abilities of CNNs [14,32–39]. Hence, we now arrive at the question: *How can we*

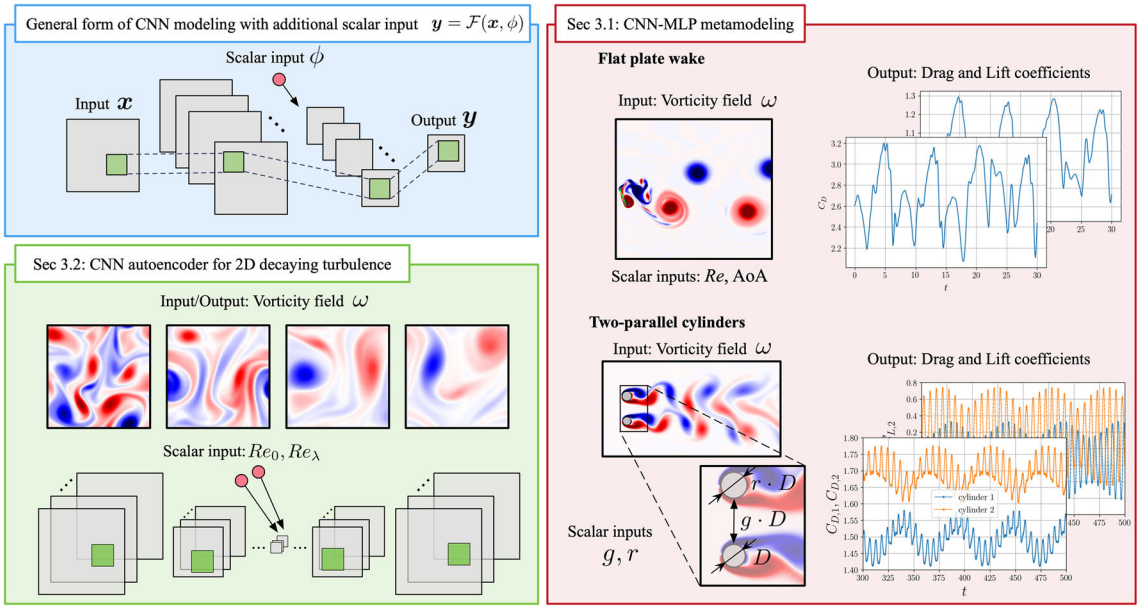

Fig. 1 Overview of the present study. We consider a general form of CNN-based modeling with additional scalar input $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \phi)$

determine these strategies to achieve a better performance of CNN and fluid flows?—In this paper, we tackle this vague portion with regard to the use of CNN and fluid flow analyses.

As mentioned earlier, we consider CNN-based regression tasks $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \phi)$ of fluid flow analyses. In this paper, we consider two types of CNN-aided modeling, as illustrated in Fig. 1: (1) CNN-MLP-based metamodeling of unsteady laminar wakes and (2) CNN-AE for turbulence, so as to investigate various considerable parameters inside CNN. The organization of the present paper is as follows: We first introduce the basic principle of CNN and the covered models based on CNN in Sect. 2.1. The information for fluid flow data used in the present paper is offered in Sect. 2.2. Results and discussion are provided in Sect. 3. We finally give concluding remarks in Sect. 4.

2 Methods

2.1 Convolutional neural network (CNN) for fluid flow analyses

2.1.1 Basic principle of CNN

Let us first introduce operations inside a convolutional neural network (CNN) [40]. The CNN was originally developed in image recognition tasks. In particular, various efforts on CNN-based image classification can now be widely seen such as ResNet [41], GoogLeNet [42], and SENet [43]. Because CNN is good at handling high-dimensional data through the concept of weight sharing, it is also becoming one of the promising tools to analyze fluid flows [44–48].

A CNN typically consists of several types of layers, i.e., convolutional layer, pooling layer, and up-sampling layer, as illustrated in Fig. 2a with an example of CNN autoencoder. A main operation of CNN is performed at the convolutional layer which extracts spatial features of input data through filter operations, as presented in Fig. 2b. A fundamental operation of convolutional layer shown in Fig. 2b is taking a summation of an Hadamard product of an arbitrary portion of input data and a filter h. Output data of a convolutional layer $q^{(s)}$ can be expressed as,

$$q_{ijn}^{(s)} = \varphi\left(\sum_{m=1}^{M}\sum_{p=0}^{H-1}\sum_{q=0}^{H-1} h_{pqmn}^{(s)} q_{i+p-G,j+q-G,m}^{(s-1)} + b_n^{(s)}\right), \tag{1}$$

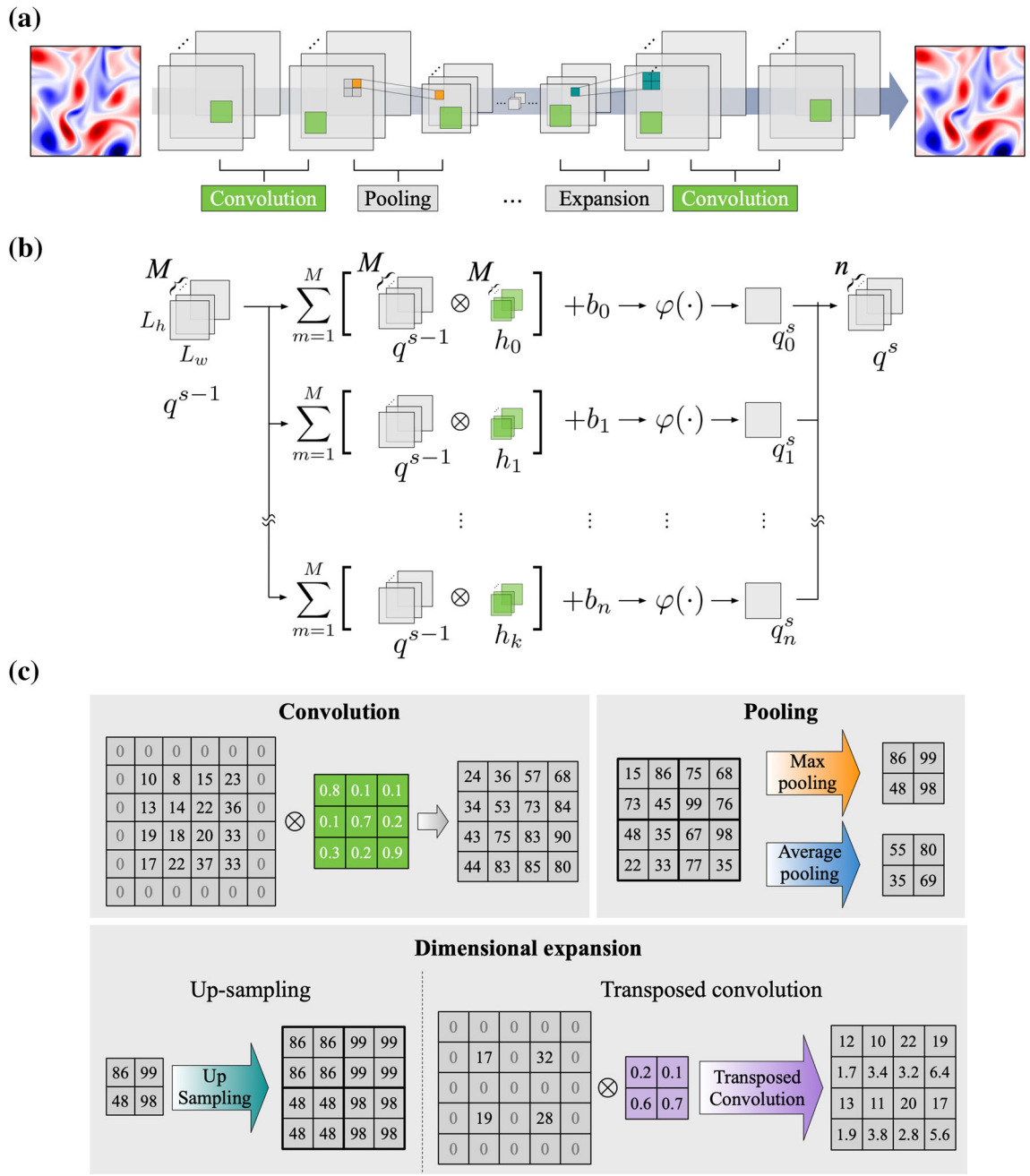

Fig. 2 Basic operations of CNN. **a** A structure of typical CNN-AE with convolutional layers, pooling layers, and dimensional expanding layers. **b** Convolutional operation at each convolutional layer to obtain q^s from q^{s-1}. **c** Convolution, pooling, and dimensional expansion. Gray zero values indicate additionally embedded values for each operation

where $G = \lfloor H/2 \rfloor$ (where $\lfloor \cdot \rfloor$ represents the operation of rounding down the value to the nearest decimal), H is width and height of the filter, M is the number of input channel, n is the number of output channel, b is a bias, and φ is an activation function, respectively. Note that we showed the two-dimensional operation above since two-dimensional flows are only handled through the present study, although its extension to three-dimensional flows is rather straightforward [29,49] albeit computationally more expensive. The nonlinear activation function φ enables a machine learning model to account for nonlinearities into its estimation. There are various choices of nonlinear activation functions [50]. We utilize the ReLU function [51] which can avoid vanishing the gradient of weights in deep CNNs. Weights $\boldsymbol{w}$ (values on filters) are optimized through a backpropagation [52] to minimize the loss function between estimated data and reference data $\boldsymbol{q}_{\mathrm{Ref}}$,

$$\boldsymbol{w} = \mathrm{argmin}_{\boldsymbol{w}} ||\boldsymbol{q}^{(s_{\max})} - \boldsymbol{q}_{\mathrm{Ref}}||_2, \tag{2}$$

where $\boldsymbol{q}^{(s_{\max})}$ is an output of CNN at the last layer $s_{\max}$.

We also incorporate other layers to process data inside the CNN. One of them is a pooling layer which downscales the data. This feature is useful in reducing data dimension for regression or classification problems—for example, to estimate a scalar value from high-dimensional data [1], and to extract key features using an autoencoder [20,23,53]. There are mainly two methods for downsampling in the pooling layer, i.e., max and average pooling, as illustrated in Fig. 2c. Both methods are common in extracting a representative value from an arbitrary area; the difference only lies in whether to extract, max, or average values. This difference can mathematically be written as,

$$q_{ij}^{\mathrm{LR}} = \left(\frac{1}{\gamma^2} \sum_{k,l \in P_{i,j}} (q_{kl}^{\mathrm{HR}})^P \right)^{1/P}, \tag{3}$$

where $P = 1$ or ∞ provides average or max pooling, for the arbitrary region ($\gamma \times \gamma$) of the input data q^{HR}.

Contrary to the pooling operation, we can also consider expanding the dimension of the data, which is required, e.g., in constructing a CNN autoencoder (for decoder part) and in estimating the two-dimensional sectional flow field from scalar values of sensor measurements using CNN [1,20,21,50,54]. There are also mainly two techniques to expand the dimension, i.e., up-sampling and transposed convolution, as shown in Fig. 2c. The up-sampling is a simple operation which copies the value onto an arbitrary region. On the other hand, the transposed convolution [55,56] is fundamentally an inverse operation of regular convolution. As illustrated in Fig. 2c, the input data are first expanded by embedding zero value among grid points. The regular convolutional operation (Eq. 1) is then applied to expand the dimension of data.

As introduced above, we have various candidates to construct a CNN depending on users' tasks. However, to the best of our knowledge, the influence of various parameters inside CNNs on their ability has not yet been investigated in detail, despite that these operations are deemed to keys for CNN-based fluid flow analyses. In this study, we address this point by focusing on the choice of downsampling operations and dimensional expanding techniques.

2.1.2 Covered CNN models

As mentioned above, we consider two types of convolutional neural network (CNN)-based architectures: 1. combination of CNN and multilayer perceptron (CNN-MLP) and 2. CNN-based autoencoder (CNN-AE), which have widely been utilized in both regression and classification tasks [57].

Most of the image classification models consist of the CNN-MLP model since it generally outputs scalar values as a probability of each class from sectional or volumetric data [58]. It is also utilized in fluid flow analyses when it is required to output scalar variables, e.g., aerodynamics coefficients, from two-dimensional data such as flow fields [35]. Considering these, the present CNN-MLP model is constructed as follows;

1. Convolutional layers and pooling layers are, respectively, utilized to extract key features and to downsample images, as shown in Fig. 3.
2. An MLP is then adopted to output scalar values after reshaping (i.e., "Flatten" in Fig. 3) the data to a one-column matrix.

In this study, we aim to estimate drag and lift coefficients of flows over two side-by-side cylinders and a flat plate.

When utilizing CNNs for coefficient estimations of fluid flows, additional scalar values which strongly relate to fluid flow phenomena, e.g., the Reynolds number, the Mach number, and the angle of attack, are often inserted to help its estimations [35,36]. Although this is a popular technique, the placement of these scalar inputs has usually been decided by users without thorough investigation to date. To investigate this point, we consider additional scalar inputs for each example,

1. the chord Reynolds number Re_c and the angle of attack α for a flat plate wake and
2. the diameter ratio of two cylinders r and distance between two cylinders g for two parallel cylinders' wake,

from four different cases of placements, as illustrated in Fig. 3. For cases 1 and 2, the dimension of scalar inputs is expanded using MLP and CNN to concatenate with the output of the first and third convolutional

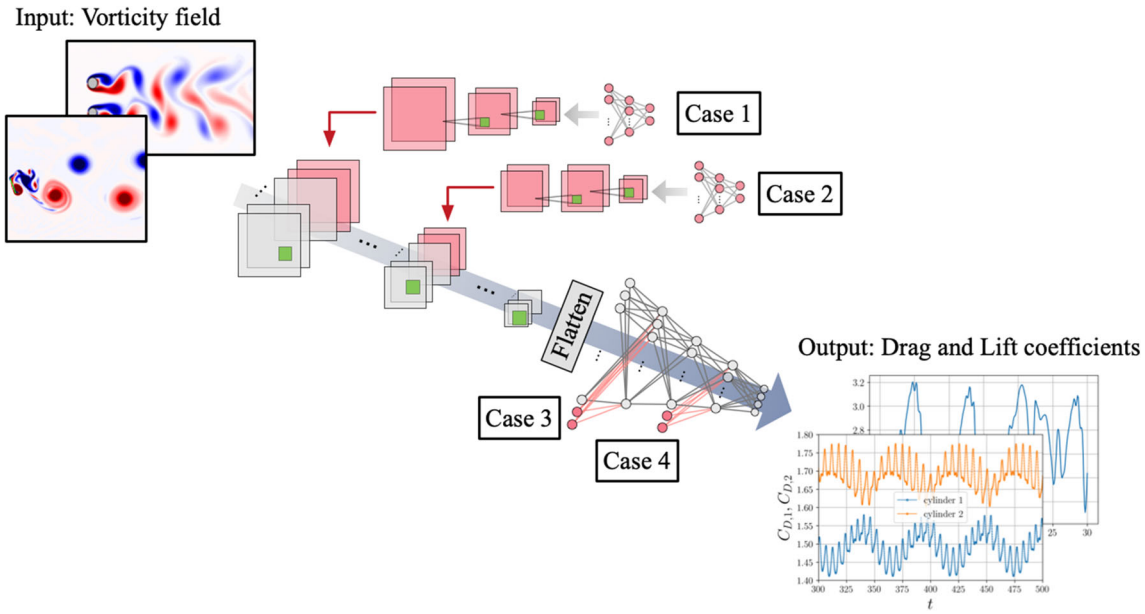

Fig. 3 CNN-MLP model covered in this study. Pink nodes indicate additional scalar inputs of the flow: 1. $\{r, g\}$ for two side-by-side cylinders flow and 2. $\{\mathrm{Re}_c, \alpha\}$ for flat plate wake

layers. In contrast, the scalar values are directly concatenated to the first and sixth layers of MLP for cases 3 and 4.

Another well-known method of CNN-based fluid flow analyses is the utilization as an autoencoder (AE). An AE [53] is used to map high-dimensional data into a low-dimensional feature space referred to as the latent vector. In particular, CNN-AE, an AE with convolutional layers, is known as a good candidate to extract key features of two-dimensional images [20,29,33,34,59]. An encoder part—the first half of CNN-AE—contains the pooling layers and the convolutional layers to extract key features of input data while reducing the dimension of data. A decoder part—the latter half of CNN-AE—then operates for expanding the dimension of the data. If we can obtain the same output as the input data through the bottleneck procedure, it implies that the information of input data can successfully be low dimensionalized into the latent space. In the present study, the up-sampling layer and the transposed convolutional layer are considered for the means of dimension expansion, as introduced in Sect. 2.1.1.

For the investigation of CNN-AE, we utilize a decaying homogeneous isotropic turbulence, as presented in Fig. 4. To concatenate the scalar values with the convolutional layers, these scalars are expanded to two-dimensional sectional data using the MLP and convolutional layers similar to the investigation of the CNN-MLP model. Here, we consider four different input placements, i.e., the input layer (case 1), the third convolutional layer (case 2), the latent space (case 3), and the 16th layer in the decoder part (case 4).

We perform a threefold cross-validation [60] for all neural networks and utilize the mean L_2 error norm to assess the ability of networks for each case. We use the Adam optimizer [52] for updating the weights, and training/validation data are randomly sampled for training. In what follows, the error assessment of CNN-MLP model is performed using test data excluded from the training data process, although the flow configuration is the same as that for training. For the autoencoder, i.e., CNN-AE, in contrast, we sample test data from the training data range.

2.2 Fluid flow data sets

2.2.1 Flat plate wake

We consider a flat plate wake at the chord-based Reynolds number of $\mathrm{Re_c} = \{100, 1000\}$ with the angle of attack $\alpha = \{35°, 70°\}$. A two-dimensional direct numerical simulation (DNS) with an immersed boundary projection method [61] is performed to simulate the flows. The solver employs a discrete vorticity-stream function formulation with an immersed boundary framework to generate the plate. The leading edge of the flat

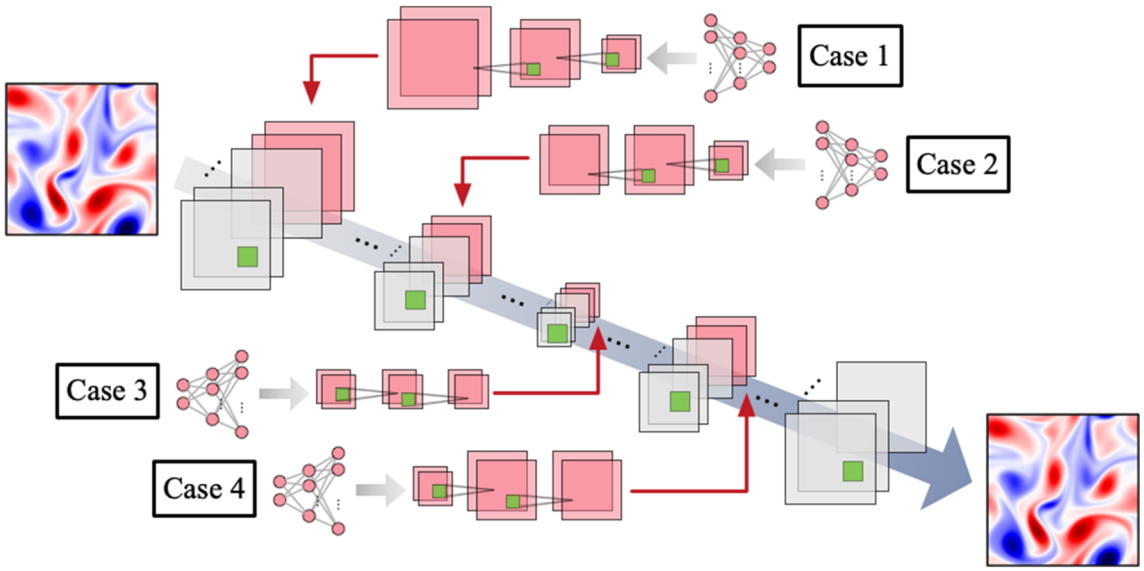

Fig. 4 CNN-AE model utilized with two-dimensional decaying turbulence. Pink nodes indicate additional scalar inputs of the flow, $\{\mathrm{Re}_0, \mathrm{Re}_\lambda\}$ (color figure online)

plate is placed at $(x/c, y/c) = (0, 0)$. A multi-domain technique [62] with five grid levels is utilized with the finest inner-most domain as $-0.2 \leq x/c \leq 1.8$, $-1 \leq y/c \leq 1$ and grid spacing of $\Delta x/c = 0.008$. The solver uses the Crank–Nicolson scheme for the viscous term and an explicit second-order Adam–Bashforth method for the advective term. At the far-field boundaries, uniform flow is prescribed. To possess sufficient spatial and temporal resolution for estimation of lift and drag forces, we collect 1200 snapshots within the spatial domain $-2.5 \leq x/c \leq 8.73$, $-4 \leq y/c \leq 4$ at a sampling frequency of $fU_\infty/c = 10$.

In all four cases analyzed, unsteady vortex shedding behavior is observed in the wake. The complexity of the vortical structures increases with the angle of attack and also with the Reynolds number, as shown in Fig. 5. As the nonlinear interactions in the wake increase, the time-averaged flow fields reveal the deviation of the vortical structures from the centerline. The time-averaged lift $\bar{C}_L$ and drag forces $\bar{C}_D$ along with the dominant shedding frequency St for each case are also summarized in Fig. 5. The drag forces increase dramatically with the angle of attack, while the shedding frequency decreases.

2.2.2 Wake of two side-by-side cylinders

The wake interactions between two side-by-side circular cylinders with uneven diameter are considered. A schematic view of the problem setup is shown in Fig. 6a. The two circular cylinders with a size ratio of r are separated with a gap of gD, where g is the gap ratio. The Reynolds number is fixed at $\mathrm{Re}_D = U_\infty D/\nu = 100$. The two cylinders are placed $20D$ downstream of the inlet where a uniform flow with velocity U_∞ is prescribed, and $40D$ upstream of the outlet with zero pressure. The side boundaries are specified as slip and are $40D$ apart. The flows over the two cylinders are solved by the open-source CFD toolbox OpenFOAM [63], using second-order discretization schemes in both time and space.

The flow physics is governed by two parameters: the size ratio r and the gap ratio g. The flow fields of $r = \{1.00, 1.15, 1.30\}$ and $g = \{0.5-2.5\}$ are shown in Fig. 6. For $g = \{0.5, 0.7, 1.0\}$, the wakes are generally chaotic for all three size ratios. For higher gap ratios, the wakes restore order, but are characterized by different features. In the case of two identical cylinders ($r = 1.00$), for $g = \{1.5, 2.0\}$, the vortices shed from the two cylinders are in phase with each other in the near wake and merge into a larger binary vortex street downstream. At $g = 2.5$, the parallel vortex streets are out of phase with each other, and they form a pair of symmetric flow patterns with respect to x axis. For cylinders with a different but close diameter ($r = 1.15$), the two vortex streets with different natural shedding frequencies can synchronize to form a single binary vortex street, as is the case for $g = 1.5$. With the increase in the gap ratio, the coupling between the two vortex streets becomes weaker. As a result, the wakes of $g = \{2.0, 2.5\}$ are characterized by quasi-periodicity due to the interactions between the two vortex streets with different shedding frequencies. Such quasi-periodic flow also prevails for

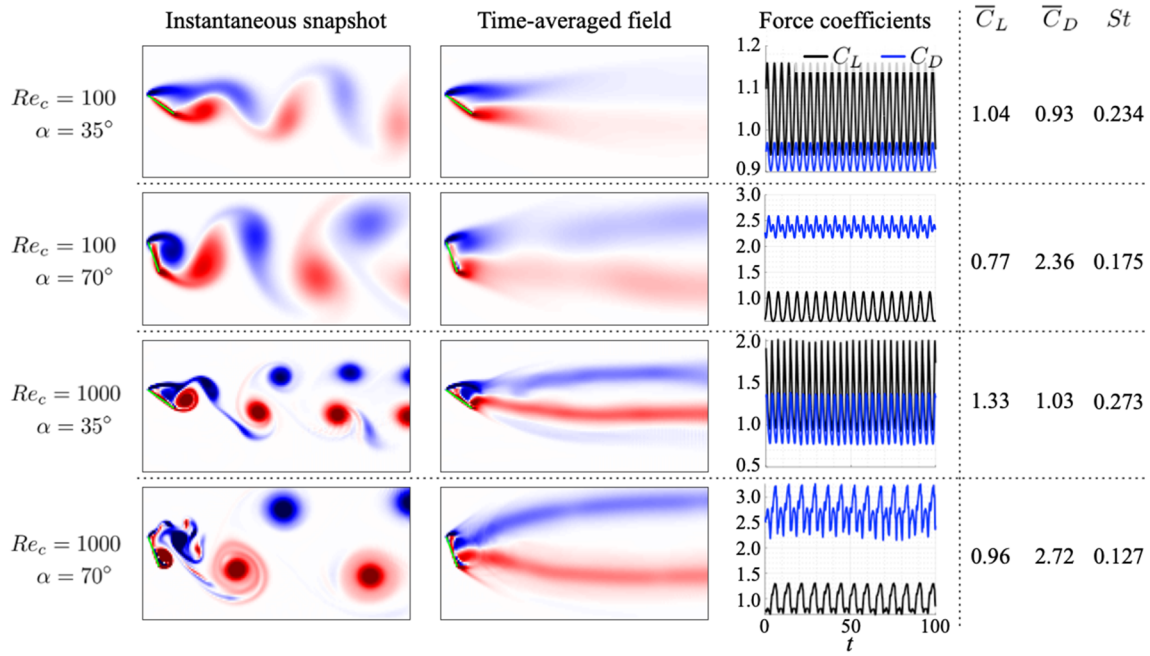

Fig. 5 Wakes of flow over a flat plate. Shown are the instantaneous (first column) and time-averaged (middle column) vorticity fields $\omega \in [-7, 7]$, with blue and red representing negative and positive values, respectively. The lift and drag coefficients for each case are shown in the last column

cases with $(r, g) = (1.30, \{1.5 - 2.5\})$, for which synchronization does not occur due to the large difference in the natural shedding frequencies.

2.2.3 Decaying homogeneous isotropic turbulence

To examine the influence of CNN performance on various parameters inside the CNN-AE, we also consider a two-dimensional decaying homogeneous isotropic turbulence. The training data set is prepared by a DNS [64]. The governing equation is the two-dimensional vorticity transport equation,

$$\partial_t \omega + \boldsymbol{u} \cdot \nabla \omega = \mathrm{Re}_0^{-1} \nabla^2 \omega, \tag{4}$$

where $\boldsymbol{u} = (u, v)$ and ω are the velocity and vorticity, respectively. The size of the computational domain is $L_x = L_y = 1$. In this study, three initial Reynolds numbers $\mathrm{Re}_0 \equiv u^* l_0^* / \nu = \{80.4, 177, 442\}$ are considered, where u^* is the characteristic velocity obtained by the square root of the spatially averaged initial kinetic energy, $l_0^* = [2\overline{u^2}(t_0)/\overline{\omega^2}(t_0)]^{1/2}$ is the initial integral length, and ν is the kinematic viscosity, as presented in Fig. 7. The numbers of grid points are $N_x = N_y = \{128, 256, 512\}$ for the covered initial Reynolds numbers $\mathrm{Re}_0 = \{80.4, 177, 442\}$, respectively. For the input and output attributes to the present AE, the vorticity field ω is utilized. Since the size of input data must be consistent over the covered Reynolds numbers to feed into the AE, the vorticity data generated at $\mathrm{Re}_0 = 80.4$ and 442 are interpolated to 256^2 size when we handle with the AE. In addition, we also use the instantaneous Taylor–Reynolds number $\mathrm{Re}_\lambda(t) = u^{\#}(t)\lambda(t)/\nu$, where $u^{\#}(t)$ is the spatial root-mean-square value for velocity of an instantaneous field and $\lambda(t)$ is the Taylor length scale of the instantaneous field, as the second scalar input, as shown in the green portion of Fig. 1. For training the present AE, we use 1000 snapshots for each initial Reynolds number in a dimensionless time of $t = 2{-}6$ with a time interval of $\Delta t = 0.004$.

3 Results and discussion

3.1 Example 1: scalar input-aided convolutional neural networks

We first assess the influence of the scalar input placements for the CNN-MLP model. As stated in Sect. 2.1.2, our consideration is the estimation of drag C_D and lift C_L coefficients of the flow over a flat plate and two

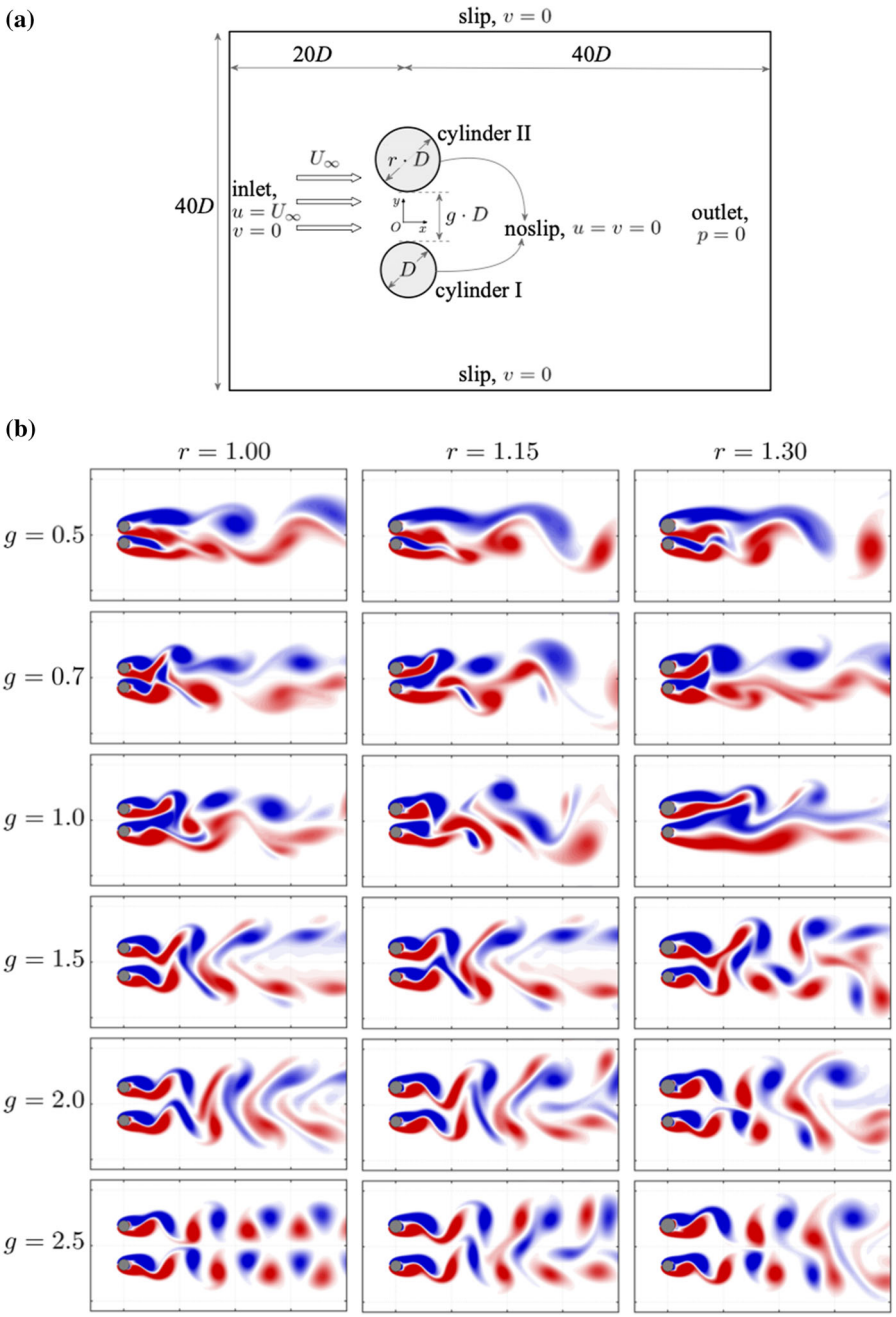

Fig. 6 **a** Computational domain for flow over two side-by-side cylinders. **b** Wakes of flow over two parallel cylinders. Shown are the vorticity fields $\omega \in [-1, 1]$, with blue and red representing negative and positive values, respectively (color figure online)

side-by-side cylinders. As already presented in Fig. 3, we investigate four input placements for feeding input scalar values, referred to as cases 1, 2, 3, and 4. In the present formulation, the dimensions of scalar inputs are expanded to concatenate with convolutional layers with cases 1 and 2, while they are directly connected to the MLP layers for cases 3 and 4. With both flow examples, a single machine learning model handles all types of flows. For instance, a single model is trained with four types of flows (combination among $\mathrm{Re_c} = \{100, 1000\}$ and $\alpha = \{35°, 70°\}$) for the flat plate wake, while other one is trained with eighteen types of flows (combination among $r = \{1.00, 1.15, 1.30\}$ and $g = \{0.5, 0.7, 1.0, 1.5, 2.0, 2.5\}$) for the two side-by-side cylinders example.

The first example for the present analysis is the flat plate wake. The L_2 error norms of the estimated drag and lift coefficients are summarized in Fig. 8a, b. Each L_2 error norm is defined as follows: $\epsilon_D =$

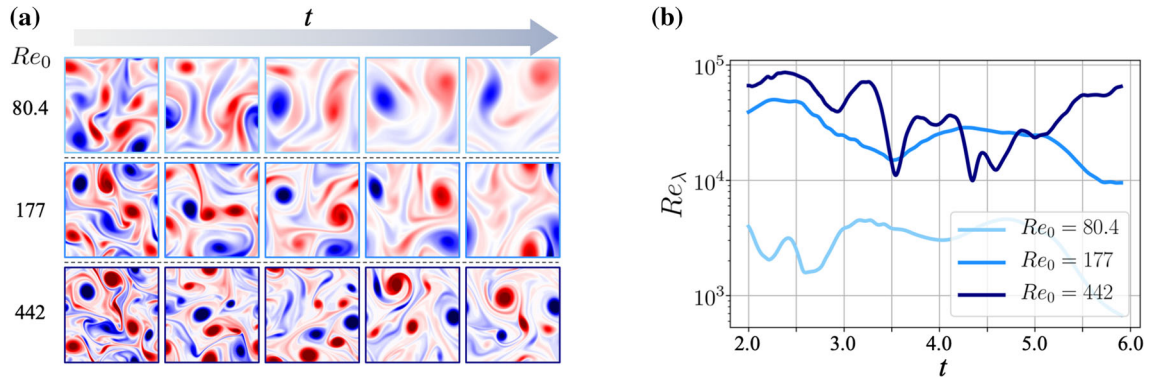

Fig. 7 Two-dimensional decaying isotropic homogeneous turbulence utilized in this study. **a** Vorticity field of flows at each initial Reynolds number Re_0 and **b** time evolution of their Taylor–Reynolds numbers Re_λ

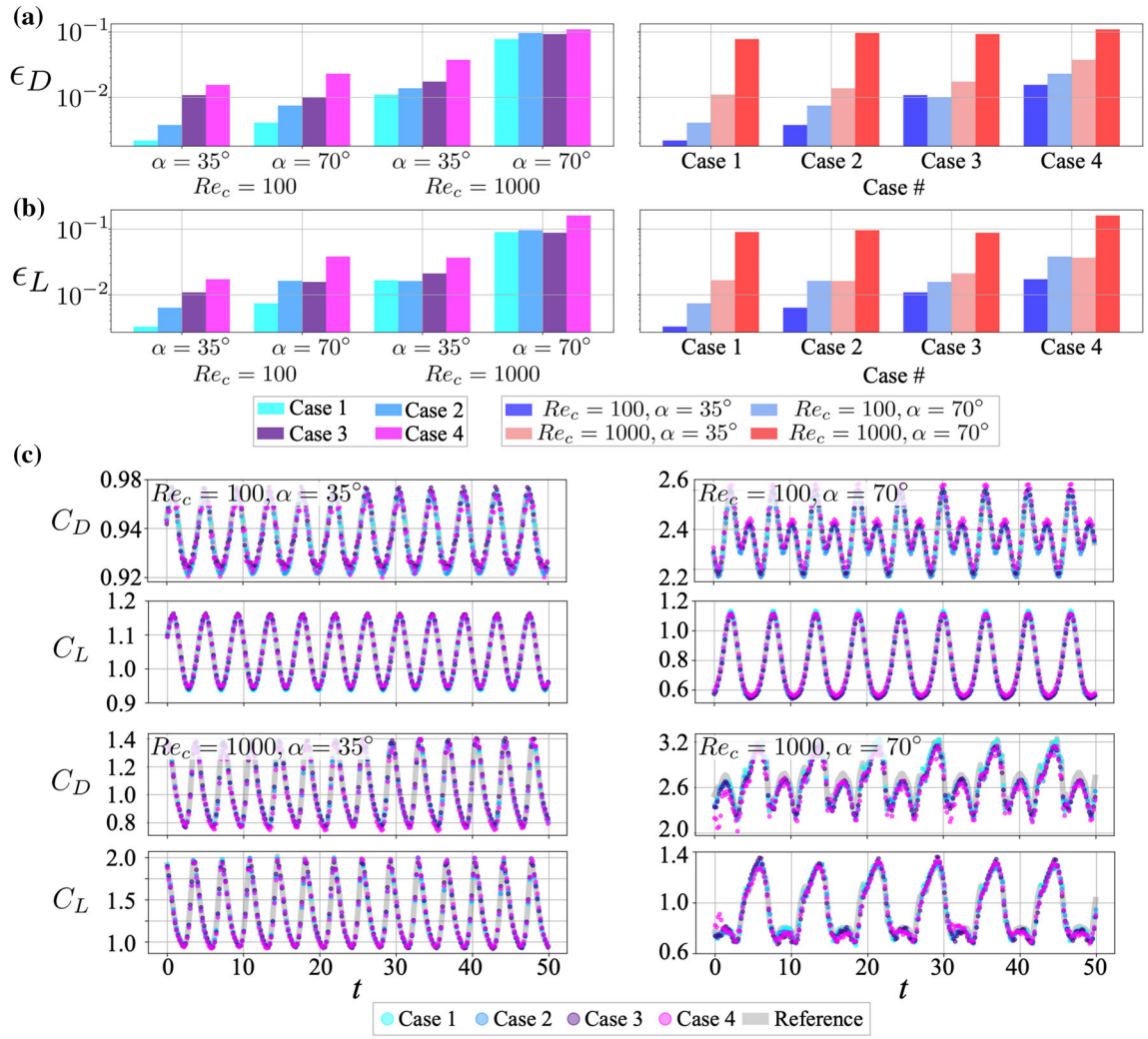

Fig. 8 Dependence of C_D and C_L estimations on input scalar placements for a flow over a flat plate. Comparison of the L_2 error norm among flow types (right side) and input placements (left side) with $\{\mathrm{Re}_c, \alpha\}$ inputs for **a** C_D and **b** C_L. **c** Time traces of estimated coefficients

Fig. 9 Dependence of the averaged L_2 error norms $\bar{\epsilon}$ of force coefficient estimations for a flow over a flat plate with $\{\mathrm{Re}_c, \alpha\} = \{1000, 70°\}$ on the number of training snapshots. The standard deviation over the cross-validation is also shown as the gray line

$||C_{D,\mathrm{Pred}} - C_{D,\mathrm{True}}||_2/||C_{D,\mathrm{True}}||_2$ for drag coefficient and $\epsilon_L = ||C_{L,\mathrm{Pred}} - C_{L,_{\mathrm{True}}}||_2/||C_{L,\mathrm{True}}||_2$ for lift coefficient, respectively. The left portion shows the relationship among the covered flow types and the error, while the right counterpart compares the cases of input placements. The basic trend here is that the estimation at higher $\mathrm{Re_c}$ and larger α is less accurate, although the highest L_2 error norm is still as low as 0.1. The L_2 error norms of C_D and C_L are almost at the same level in this case. It is striking that the cases with scalar inputs at the upperstream layers tend to show a better estimation than those with the scalar inputs at the downstream layers. This is likely because feeding the scalar inputs from the upperstream layer promotes the model's robustness for test data by merging biases and nonlinear activation functions inside the machine learning model. This trend can also be observed from the time series of estimated coefficients as shown in Fig. 8c. The result of case 4 (scalar inputs at downstream layer) shows slight disagreement with the reference data especially for the flow at $(\mathrm{Re}_c, \alpha) = (1000, 70°)$.

We also investigate the influence of the amount of training snapshot in this problem. We here consider six numbers of training snapshots $n_{\mathrm{snapshot}} = \{50, 100, 200, 300, 500, 1000\}$. The averaged L_2 error norms of C_D and C_L values for a flat plat wake with $\{\mathrm{Re}_c, \alpha\} = \{1000, 70°\}$ are shown in Fig. 9. We also present the error variance over the cases 1 to 4 for each number of snapshots as the gray line. As the number of training snapshots increases, both the overall error and the variance tend to decrease in all cases. Although we can generally see the better estimation by adding scalar inputs at the earlier layers, there are also some cases where this argument looks invalid (e.g., cases 1 and 2 at $n_{\mathrm{snapshot}} = 300$). However, we should note that the difference in the L_2 error between such cases is $\mathcal{O}(10^{-2})$, and these cases already exhibit acceptable accuracy. In addition, as we will discuss in Fig. 11, the trend that the earlier input leads to the better accuracy can be stated more clearly when the problem setting is not too simple.

Next, we consider a flow over two side-by-side cylinders to investigate the influence on the scalar input placements. As mentioned above, we utilize the diameter ratio r between cylinders I and II and the distance between two cylinders g as the additional inputs such that three cases for $r = \{1.00, 1.15, 1.30\}$, and six cases for $g = \{0.5, 0.7, 1.0, 1.5, 2.0, 2.5\}$. In this example, a single model trained with 18 cases estimates drag and lift coefficients for both cylinders $\boldsymbol{y} = (C_{D,1}, C_{D,2}, C_{L,1}, C_{D,2}) \in \mathbb{R}^4$ from the vorticity field ω.

The L_2 error norms of the estimated four coefficients, C_D and C_L for both cylinders, $\{\epsilon_{D,1}, \epsilon_{D,2}, \epsilon_{L,1}, \epsilon_{L,2}\}$ are summarized in Fig. 10. The error rate becomes higher for the flows at g lower than 1.0, corresponding to chaotic flows, contrary to the flows at higher g, i.e., periodic and quasi-periodic flows. In contrast to the example of flat plate wake, the L_2 error norms of C_L tend to be higher than that of C_D. This is likely because of the difference in magnitude—the value of C_L has the order of $\mathcal{O}(10^0)$, while that of C_D is $\mathcal{O}(10^{-1})$. More concretely, a tiny error, e.g., $\mathcal{O}(10^{-2})$, cannot be assessed fairly for C_D and C_L since its relative magnitude for them should be different from each other. To avoid this issue, standardization or normalization of the data can be a good candidate [65]. Noteworthy here is that the dependence of the CNN performance on the input placements of scalar values is lower than that with the flat plate examples. This suggests that users should care the input placement depending on the data that they handle.

The robustness against noisy input and its dependence on input placements are also examined with the two side-by-side cylinders example, as shown in Fig. 11. A Gaussian noise is added to the input vorticity field. We consider three magnitudes of the noise whose signal-to-noise ratio (SNR) is set to $\mathrm{SNR}^{-1} = \{0.012, 0.11, 0.31\}$, where $\mathrm{SNR} = \sigma^2_{\mathrm{Data}}/\sigma^2_{\mathrm{Noise}}$. In Fig. 11, the bars indicate the L_2 error norm averaged over C_D and C_L of both two cylinders for all studied values of distance factor g. (The result for each configuration is shown independently in "Appendix B.") As mentioned above, there is no clear difference among cases without noise, presented as the bright-green color bars. On the other hand, the cases with the strong magnitude of noise, i.e., blue colored bars, show notable differences over the input placements. Case 1 (i.e., inputs the scalars at the 1st convolutional layer) reports the least L_2 error norm, which is the same trend as that in the flat plate wake example. This also supports our observation above that the scalar inputs from the

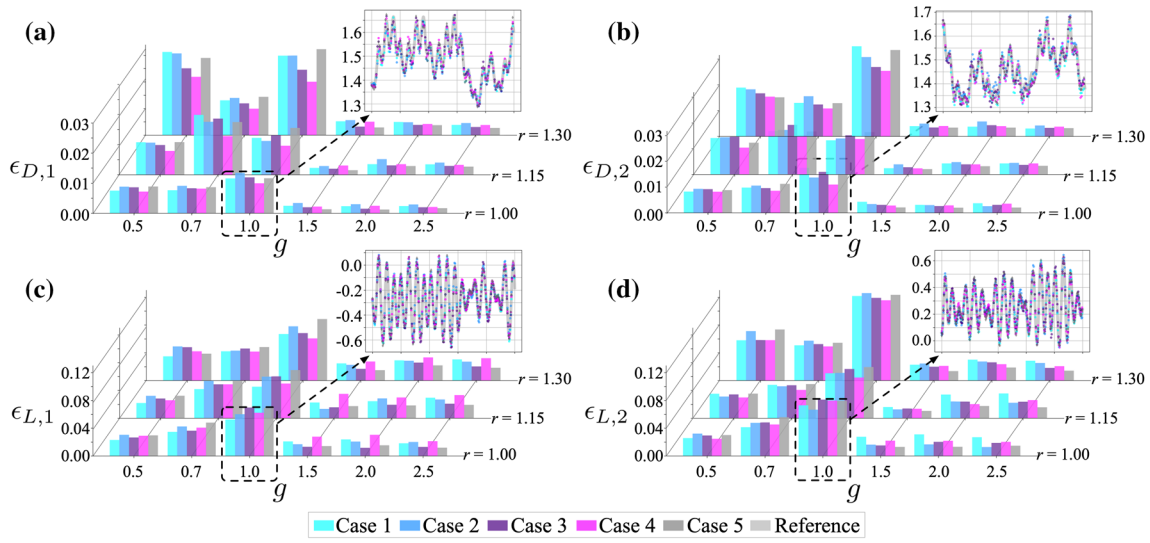

Fig. 10 Dependence of C_D and C_L estimations on input scalar placements for a flow over two side-by-side cylinders. Comparison of the L_2 error norm among flow types for each coefficient: **a** C_D of cylinder I, **b** C_D of cylinder II, **c** C_L of cylinder I, and **d** C_L of cylinder II. Case 5 denotes the model without the additional scalar value input. Time traces of estimated coefficients with $g = 1.0$ for each coefficient are also shown

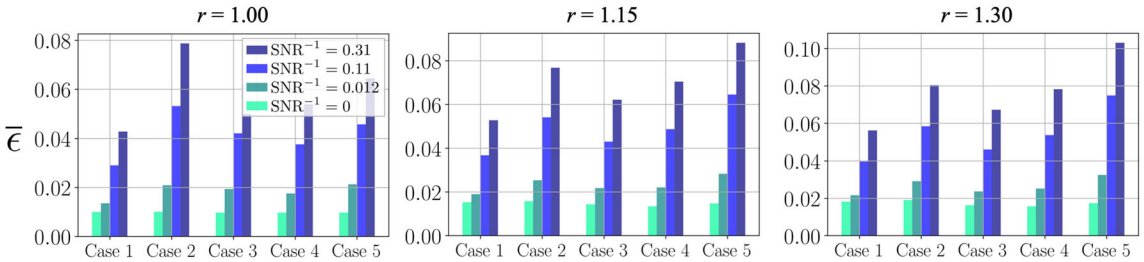

Fig. 11 Robustness against noisy input for a flow over two side-by-side cylinders. The error $\overline{\epsilon}$ represents the averaged L_2 error norm over C_D and C_L of both two cylinders for all studied values of distance factor g

upperstream layer enable the model to be robust by merging biases and nonlinear activation functions inside models. Summarizing above, care should be taken in determining the scalar input placement by considering not only the accuracy for the clean data but also the robustness against noise.

3.2 Example 2: investigation of CNN parameters for fluid flow analyses with autoencoder

In this section, let us investigate the influence on CNN parameters for fluid flow analyses using CNN-AE. Here are the parameters we focus in this study;

1. Size of the filter for the convolutional operation (Sect. 3.2.1).
2. Compression rate of CNN-AE (Sect. 3.2.2).
3. Padding operation for the convolutional operation (Sect. 3.2.3).
4. Input placements of supplemental scalar value (Sect. 3.2.4).
5. Methods to reduce/expand the dimension of the data (Sect. 3.2.5).
   (a) Up-sampling versus transposed convolution (dimension expansion)
   (b) Max versus average pooling (dimension reduction)

Hereafter, we use a vorticity field of two-dimensional decaying homogeneous isotropic turbulence as the input and output attribute of the AE. Since it contains various scales of structures and also the data are generated with the biperiodical boundary condition, it can be regarded as a suitable example for us to establish a benchmark investigation of CNN parameters, such as filter size, padding operation, and so on.

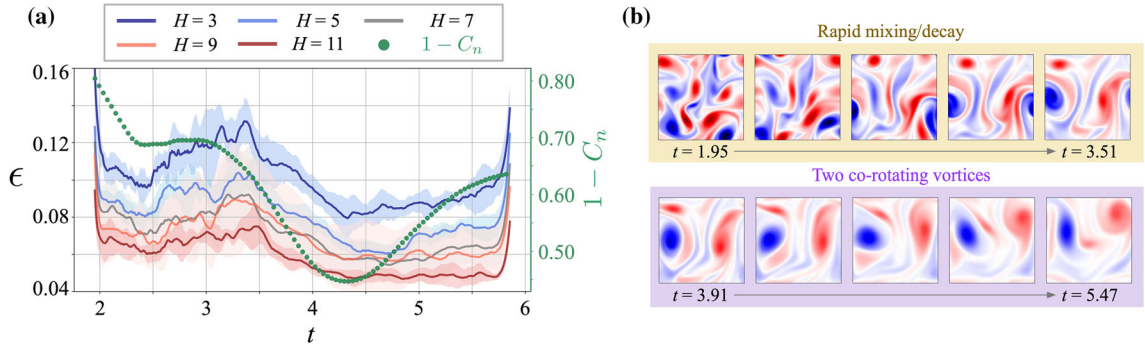

Fig. 12 **a** Dependence of the L_2 error norm on the size of the filter size of CNN-AE. Time trace of the L_2 error norm and the averaged cosine similarity between each snapshot and the other snapshots are shown. The shaded region represents the standard deviation over a threefold cross-validation. **b** Representative flow fields of handled dataset are shown. While a rapid mixing and decaying process can be observed for the flows at $1.95 < t < 3.51$, less spatial variation can be seen on the latter part of the decaying nature

3.2.1 Size of the filter for the convolutional operation

One of the most advantageous features of CNN is the filter operation which can extract key spatial structures of the input data to establish a relationship between the input and output data. Because the size of the filter directly relates to the number of weights inside CNN, we can easily expect that an appropriate choice for the filter size can lead to an improvement in estimation ability. This feature is advantageous also in fluid mechanics applications—in fact, Fukami et al. [12] utilized a customized CNN, which contains multiple sizes of filters, for fluid flow super-resolution analysis and reported its great ability in handling turbulence compared to a CNN with a single type of filter size. Otherwise, Lee and You [31] also capitalized on a CNN which includes multiple sizes of filters for the construction of surrogate modeling for high-fidelity simulation. Despite that the size of the filter plays a crucial role in CNN-based modeling, the dependence on the filter size for fluid flow data has not been investigated clearly yet. Here, we consider five sizes of filter $H = \{3, 5, 7, 9, 11\}$ and investigate its influence on the ability of CNN-AE. For this investigation, the training data are generated at $\mathrm{Re}_0 = 80.4$, and the size of the latent vector is set to be 1024.

The time series of L_2 error norm of the reconstructed field is shown in Fig. 12a. The model with the larger size of the filters shows lower error level over the time. This is simply because the number of weight increases for the larger size of the filters and eases the problem setting for the machine learning model. Along with the time series of the L_2 error norms, we also present the averaged cosine similarity in order to seek for the relationship between the variance over the training snapshots and the reconstruction error of the autoencoder. We define the averaged cosine similarity for each snapshot C_n as

$$C_n = \frac{1}{N} \sum_m \frac{\boldsymbol{\omega}_n \cdot \boldsymbol{\omega}_m}{||\boldsymbol{\omega}_n||\, ||\boldsymbol{\omega}_m||}, \tag{5}$$

where $\boldsymbol{\omega}_n$ represents the vorticity field at a certain snapshot and $\boldsymbol{\omega}_m$ represents the vorticity field to be compared with $\boldsymbol{\omega}_n$, respectively. The total number of snapshots is denoted as N. This measure enables us to investigate the influence of the structural and statistical similarities of flow fields over training data on the low-dimensionalization performance. Note that we present $1 - C_n$ so that the trend looks similar to that of the error; $1 - C_n = 0$ means that the snapshot n is completely the same as the other snapshots, $1 - C_n = 1$ indicates the orthogonality, and $1 - C_n = 2$ refers to the same structure with the opposite sign. As shown in Fig. 12a, the time trace of the error basically matches with that of the cosine similarity of each snapshot $1 - C_n$. This result suggests that the autoencoder shows the better performance for the case when the structure is similar to those in the other snapshots. For example, as shown in Fig. 12b, a rapid mixing and decaying process can be seen in the flow fields at $t < 3.51$. Since each snapshot in this period is relatively "unique" among the handled data, i.e., less averaged similarity to the other snapshots, the autoencoder shows less ability in low dimensionalizing the flow fields. In contrast, since the flow fields after $t = 3.91$ basically contain common structures of two co-rotating vortices, the averaged similarity among the data is relatively high, which leads to the better performance of the autoencoder. Hence, the training data arrangement and its sampling process affect the error in this particular example associated with the decaying nature. Moreover, including more snapshots

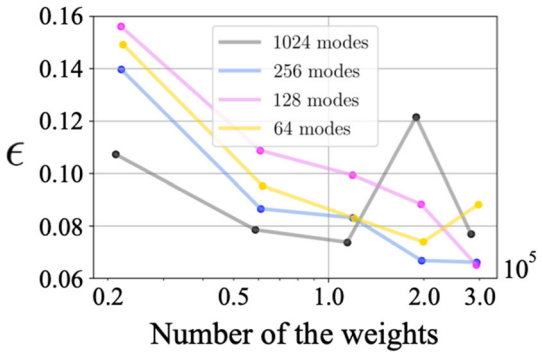

Fig. 13 Dependence of the L_2 error norm on the number of latent modes and the number of weights for CNN-AE with a two-dimensional decaying turbulence

around the beginning and the end of the time series may help to reduce the error for a time period where snapshots tend to be less similar among training data [29].

3.2.2 Compression rate of CNN-AE

We then investigate the dependence of the compression rate of the CNN-AE on the reconstruction ability. Generally, the reconstruction performance of AE becomes lower as the number of latent modes decreases [50]. Since this is a well-known fact and intuitive for us, we also assess this point with regard to the number of weights inside the CNN-AE, in addition to the number of latent modes, as shown in Fig. 13. We here present the relationship between the L_2 error norm of the reconstructed flow and the number of weights among four cases of latent modes $n_r = \{64, 128, 256, 1024\}$. Since we can increase or decrease various parameters inside CNN including the size of the filter and the number of the layers, we can have the variation for the number of weights over the same numbers of latent modes. Here, the number of weights is adjusted by changing only the size of the filters H. For the models with the number of weights less than 10^5, the error decreases as the number of weights increases, as expected. However, the trend varies for the models with the number of weights larger than 2×10^5. This is likely caused by the fact that an optimization process for weights inside a neural network becomes unstable when the number of weights is massive, which is well known as "curse of dimensionality." [66] This result encourages users to be careful in setting the number of weights to avoid the unstable learning process, in addition to the number of latent modes.

3.2.3 Padding operation for the convolutional operation

As introduced above, we have recently been able to see various studies of CNN and fluid flows. In those studies, however, we often see a suspicious setting for fluid flow analyses—a zero padding operation at convolutional layers. Fundamentally, the convolutional operation usually reduces the data size $L_w \times L_h$ to $(L_w - 2\lfloor H/2 \rfloor) \times (L_h - 2\lfloor H/2 \rfloor)$, but we often do not want the data size to be reduced via convolutional operations because the dimension reduction using pooling layers is known as the better way to acquire the robustness for spatial sensitivity and noise [67]. To avoid the dimensional reduction at the convolutional layers, it is very common to add zero values around the data (called *zero padding*), as shown in Fig. 14a. By giving additional values around an image, the dimension of the data can be retained through the convolutional operation. Although this zero padding is a commonly used technique, it is questionable from the perspective of boundary conditions, especially with numerically generated data. For instance, it can be easily suspected whether we can use the zero padding for the present two-dimensional decaying turbulence having the spatial biperiodic condition, or not. Here, we compare three types of padding operations: 1. zero padding, 2. replication padding, and 3. periodic padding. As illustrated in Fig. 14a, the replication padding is a mirroring operation which embeds symmetrical value regarding the boundary of the data. This operation is also non-physical similarly to zero padding, although the boundary values become smooth. The other covered operation is the periodic padding, which faithfully follows the boundary condition of the present turbulence data.

We investigate these three operations with various sizes of the filter H from 3 to 11, as presented in Fig. 14b. No clear difference is observed up to $H \leq 9$, while the error variance becomes significantly unstable using the replication padding with $H = 11$ as can be seen from its standard deviation shown with a shaded region. This

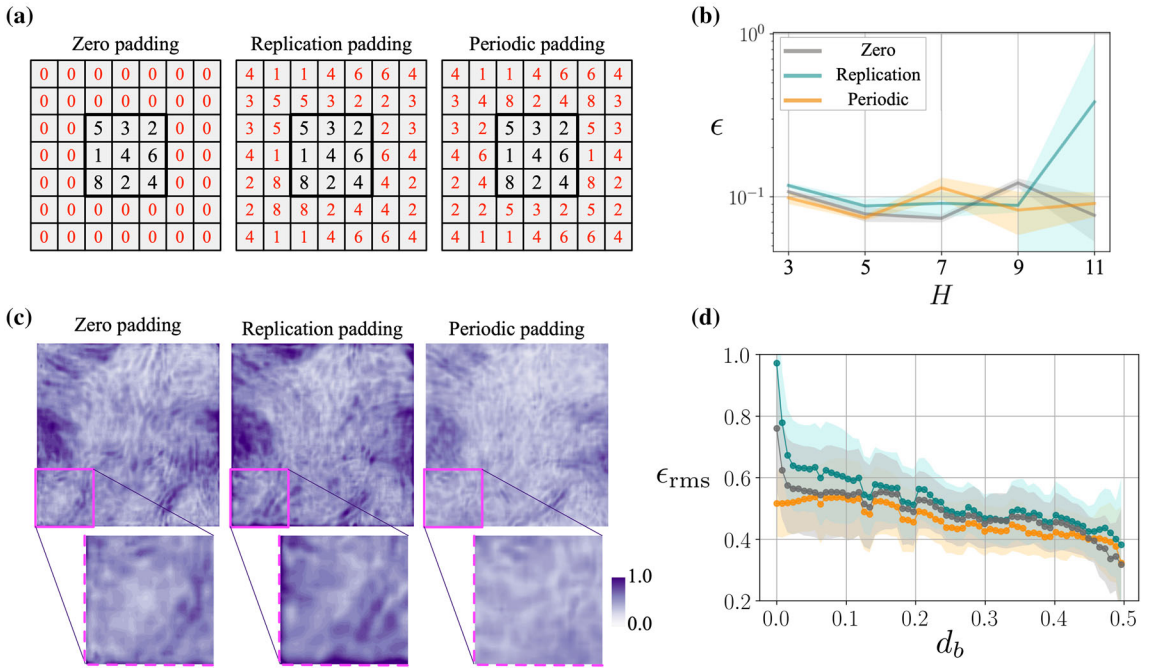

Fig. 14 **a** Padding operation covered in this study. **b** Relationship between the padding operations and the L_2 error norm for various filter sizes. The shaded region here represents the standard deviation over a threefold cross-validation. **c** The distribution of root-mean-squared L_2 error norm of each padding operation with its filter size of $H = 3$. **d** Relationship between the distribution of ϵ_{rms} and distance from boundaries d_b. The ensemble-averaged value is shown as a plot, while the shaded region represents the standard deviation of the error distribution at each d_b

negative influence of replication padding with the larger size of filters can be regarded as natural because more non-physical values are given, which suggests that these values would affect the convolutional operation inside the CNN. In contrast, what is striking here is that the models with zero padding show the same error level as those with periodic padding. This is likely because the influence of inserting zero values near the boundaries is much less than that with non-physical values by the replication padding in the operation of Eq. 1.

Let us also examine the local errors in the region near the boundaries. The distribution of root-mean-squared local L_2 error norm ϵ_{rms} with each padding operation is shown in Fig. 14c. We use the filter size of $H = 3$, as an example. While the error magnitude of replication padding is relatively larger at the region near the boundaries (shown as pink dotted lines), we can avoid the boundary influence by utilizing the zero padding and the periodic padding.

We also assess the dependence of the local ϵ_{rms} on the distance from the nearest boundary d_b. Since the present domain is a square box with $L_x = L_y = 1$, the distance d_b can simply be defined as

$$d_b = \min(d_x, d_y), \tag{6}$$

where

$$d_x = \min(x, 1 - x), \ d_y = \min(y, 1 - y). \tag{7}$$

The relationship between d_b and ϵ_{rms} is presented in Fig. 14d. In the figure, the ensemble average over the points having the same d_b is shown as a plot, while the shaded region represents the standard deviation. In the proximity of the boundaries, i.e., $d_b \simeq 0$, the error rate jumps to $\epsilon_{\mathrm{rms}} \simeq 1.0$ for the case of the replication padding, which is the largest among the covered cases. The error rate of the zero padding also increases at $d_b \simeq 0$; however, the error rapidly (at $d_b > 0.05$) decreases to the almost same level as that with periodic padding. Concluding the discussion above, the optimal choice of padding operation may differ among flows with different boundary conditions.

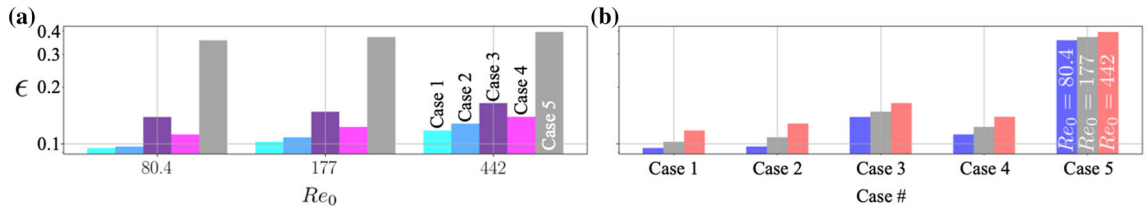

Fig. 15 Dependence of supplemental scalar value $\{\mathrm{Re}_0, \mathrm{Re}_\lambda\}$ on input placements for the CNN-AE with two-dimensional decaying turbulence. **a** Relationship between the L_2 error norm and initial Reynolds number depending on input placement. **b** Relationship between the L_2 error norm and input placement depending on initial Reynolds number

3.2.4 Input placements of supplemental scalar value

As introduced above, the supplemental scalar values are often utilized to improve the estimation ability of CNN for fluid flow analyses. Similar to the investigation for the influence on input placements of scalar values with the CNN-MLP model in Sect. 3.1, let us also examine this point with the CNN-AE for the decaying isotropic homogeneous turbulence. As briefly explained in Sect. 2.1.2, we consider four cases of input placements for additional scalar values, i.e., the 1st convolutional layer (case 1), the 4th convolutional layer (case 2), the latent space (case 3), and the 16th convolutional layer (case 4). As illustrated in Fig. 4, the scalar values are first expanded with MLP and then reshaped into two-dimensional sectional data to concatenate with outputs provided at the hidden layer inside the CNN-AE. As supplemental scalar values, we utilize two types of the Reynolds number: the initial Reynolds number Re_0 and the instantaneous Taylor–Reynolds number Re_λ. While the initial Reynolds number Re_0 represents the initial condition and governs the decaying nature of flows, the instantaneous Taylor–Reynolds number Re_λ contains information of an instantaneous flow snapshot. In addition, we also consider the case with no additional scalar values as case 5.

The dependence of the L_2 error norm of reconstructed flows on the input scalar placements is summarized in Fig. 15. As we can clearly see, case 5 (without the scalar inputs) reports higher error than other cases. This implies that the use of supplemental scalars is beneficial also for low dimensionalization. For the cases with scalar inputs, case 1 (input at the first layer) shows the best estimation among the covered cases, which is analogous to the investigation with the CNN-MLP model above. On the other hand, case 3 (input from the latent space) reports the worst estimation. This is likely due to the difference in the number of weights given at the MLP part for merging the Reynolds number input and the main part of CNN-AE. Summarizing above, the results indicate that scalar inputs at upstream layers help the estimation for CNN-AE, as well as the CNN-MLP model, and additionally, a sufficient number of weights are required before merging the input scalars with CNN-AE.

3.2.5 Dimensional reducing/expanding methods

Finally, we investigate the effect of the dimensional reduction and expansion methods for CNN performance. The relationship between the L_2 error norm and the dimension reduction ways is summarized in Fig. 16a. We compare the max and average poolings, as introduced in Sect. 2.1.1. As an example, the number of latent variables is set to be 128 with five cases of filter size $H = \{3, 5, 7, 9, 11\}$. As shown, the error with both methods is approximately at the same level for every case. This result indicates that the present AE is not very sensitive to the pooling operations.

We also check the influence of dimensional expansion methods, as presented in Fig. 16b. As introduced in Sect. 2.1.2 and Fig. 2, the significant difference between the transposed convolution and the up-sampling is whether there are trainable weights (filter) or not. Therefore, the number of weights inside a model n_{weight} increases when up-sampling layers are replaced with transposed convolutional layers, as shown by the pink curve in Fig. 16b.

As summarized here, the L_2 error norm of the models with up-sampling operation gradually decreases as the filter size increases. This exactly corresponds to the increase in the number of weights. In contrast, the variance of the model with the transposed convolutional operation is significantly unstable with $H \geq 9$ where the number of the weights reaches approximately 3.0×10^5. This is analogous to the observation in Sect. 3.2.2, known as "curse of dimensionality."

The unsuccessful training procedure of the network with transposed convolution can also be found with the appearance of checkerboard artifacts [68]. The checkerboard artifacts tend to appear with strided/transposed

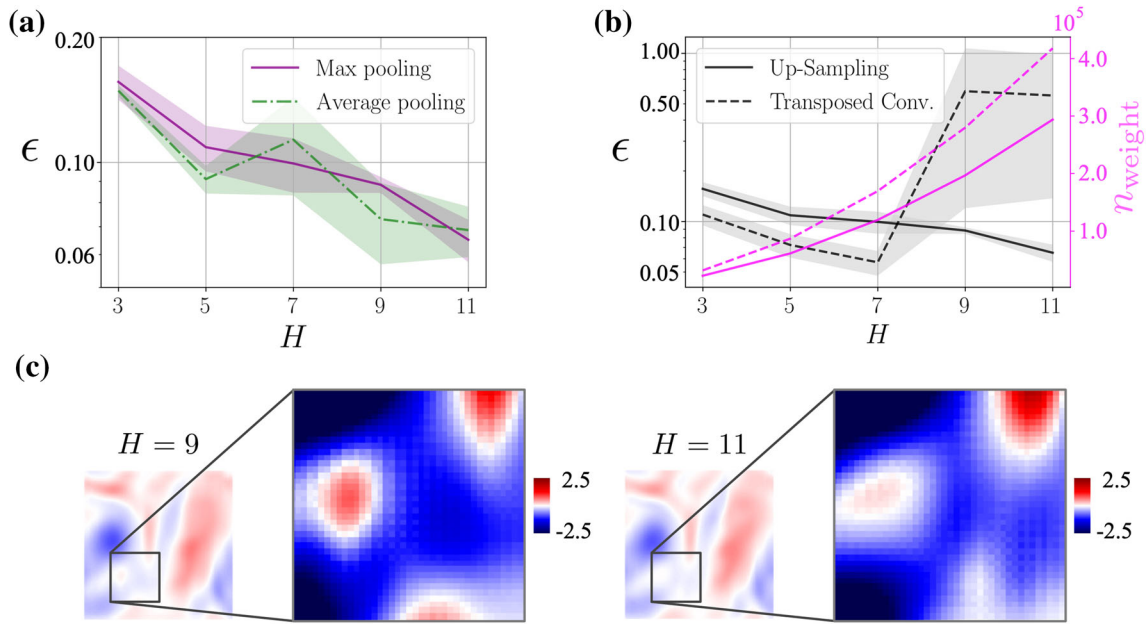

Fig. 16 Relationship between the L_2 error norm ϵ and filter size H for **a** dimension reduction methods (max and average pooling), and **b** dimension expansion methods (up-sampling and transposed convolution). The shaded region represents the standard deviation over a threefold cross-validation. **c** The estimated fields of models using transposed convolution with the filter size of $H = 9$ and 11 are shown. A checkerboard artifact can be seen due to the unsuccessful training by using the transposed convolution with the large filter size

convolution, especially when the network fails to update the weights properly through its training procedure. In our cases, the fields estimated by the networks with the filter size of $H = 9$ and 11, which reported significantly worse L_2 error norm in Fig. 16b, exhibit such artifacts, as shown in Fig. 16c. This is again due to the excessive number of weights contained in the network and a sign that the transposed convolutional operations with a relatively large filter size negatively affect the training procedure. We note that the present model with the transposed convolution generally shows better performance with less amount of weights, i.e., $H \leq 7$. Hence, the choice for the dimension expansion and reduction methods inside CNNs should be cared depending on considered flows and their model configurations.

4 Conclusions

We investigated the applicability of convolutional neural networks (CNNs), which have been utilized as a powerful tool in scientific machine learning, for fluid flow analyses. Particularly, we focused on the CNN application with additional scalar input information such that $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \phi)$. Capitalizing on the canonical fluid flow data with the perspectives on metamodeling for aerodynamics characteristics and low dimensionalization, we attempted to clear vague portions of CNN and fluid flow analyses as graphically summarized in Fig. 17.

We first considered the CNN-MLP model with additional scalar inputs, by considering flows around an inclined flat plate and two side-by-side cylinders. The model attempted to estimate drag and lift coefficients from a vorticity field. The supplemental scalar values were added to the model at four different placements, and we investigated the influence on the input placements for the scalar values. Although the response of the machine-learned model depends on target flows and associated problem settings, we observed a general trend that better estimations can be attained by placing the scalar inputs at earlier layers, not only in terms of accuracy but also robustness, especially under relatively tough problem settings, i.e., less amount of training data and noisy input. This is likely because a neural network gains robustness against the test data by merging supplemental information with biases and nonlinear activation functions inside the neural network. Although our investigation was performed using the same flow configuration as that used for training, it is widely known that neural networks can acquire the robustness against unseen flows by preparing a proper training data set [19,33,34]. We believe that our present findings can also be useful by unifying these previous studies related to the applicability to unseen flow data.

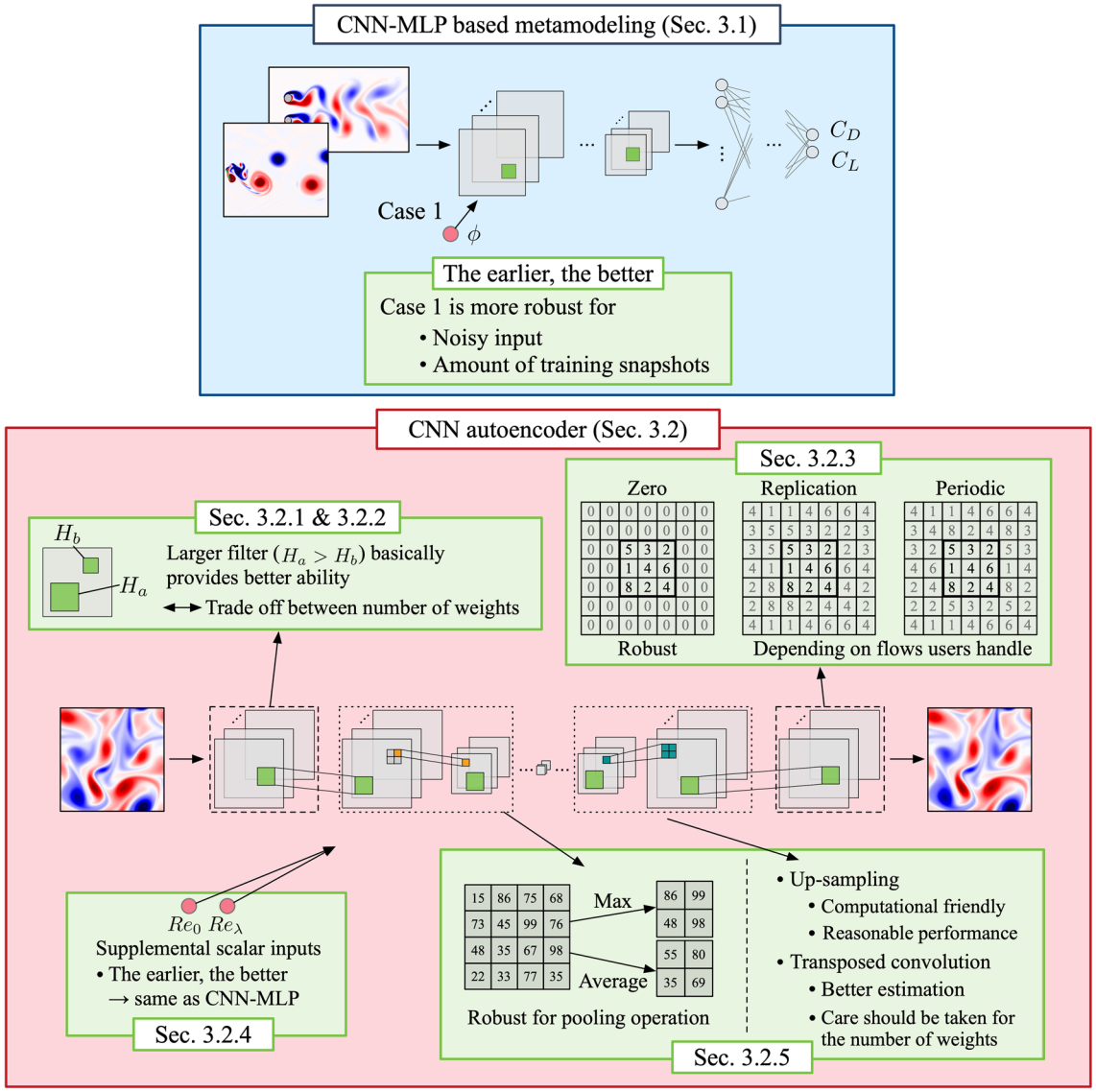

Fig. 17 Graphical conclusion of the present study

We also examined the influence on CNN parameters considering an autoencoder (AE)-based low dimensionalization. Through the investigations for the filter size and the compression rate dependence, we found that the larger number of weights contained in the model can lead to the better estimation. However, care should be taken for cases which contains too many weights since it causes unstableness of the training procedure of CNN. The dependence on the padding operation for convolution was also examined in a quantitative manner. Although the periodic padding shows the best performance since the present decaying turbulence is simulated under the biperiodic condition, the zero padding is also found to give a reasonable overall accuracy. Users can choose the optimal padding operation depending on the boundary conditions of flow data. Similar to the investigation on the CNN-MLP models, we also checked the influence of input placements for supplemental scalar values on the estimation ability of CNN-AE. Here, we also observed that the scalar input at earlier layers leads to improve the estimation more compared to the other cases. Finally, the choice of dimensional reduction/expansion methods was also considered. The machine learning model is less sensitive to the dimensional reduction methods, while it is sensitive to the expansion methods, i.e., up-sampling and transposed convolution. Since the model with the transposed convolution contains large amount of weights than that with up-sampling

layers, the model may fall into the "curse of dimensionality." Based on the above, users can attempt to choose an appropriate combination of weights depending on their problem setting and computational environment.

The present investigations enable us to notice some remaining issues and considerable extensions of CNN and fluid flow analyses. One of them is applications of CNN to unstructured mesh fluid flow data. As expressed in Sect. 2.1.1, the basic principle inside CNN is taking filters for fluid flow *image* handled on structured/discretized grids arranged with a *uniform* manner. However, as readers have already noticed, we often encounter unstructured data for many fluid flow analyses, e.g., flows around an airfoil at practical Reynolds numbers. Unfortunately, the conventional CNN used through this study cannot be applied directly to these data on unstructured mesh. To overcome this issue, several ideas have recently been proposed, e.g., graph convolutional neural networks [69,70], PointNet [71], PhyGeoNet [72], and Voronoi tessellation-aided CNN model [73]. We can expect that the present knowledge and the ways for analyzing CNN parameters with fluid flows can be extended to the aforementioned models. Otherwise, readers' interest in a practical manner must arrive at the perspective on the interpretability of results provided by CNNs. For real-world applications, we often desire grounds of estimation by a model (not only machine learning). Actually, some studies have tackled this point from various views, e.g., the relationship between machine-learned results and vortical motion [44], uncertainty quantification [74], and visualization inside CNNs [19]. In addition, the applicability of a supervised machine-learned model for test situations which are completely different from the training data regime is also one of the key factors toward practical applications [75]. Although it is still an ongoing area of research in fluid mechanics and CNN, these trends may be a good direction toward practical applications.

Acknowledgements Masaki Morimoto, Kai Fukami, and Koji Fukagata were supported by JSPS KAKENHI Grant Number 18H03758, 21H05007. Masaki Morimoto and Kai Fukami thank Mr. Taichi Nakamura (Keio University) for insightful discussion. Aditya G. Nair acknowledges the support by UNR VP Research Startup PG19679.

Declarations

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Appendix A: Details of neural networks

Here, we provide the detailed information on the neural networks covered in this study. In Tables 1 and 2, we provide the structure of the CNN-MLP model for force coefficient estimation. The main stream of the neural networks is shown in the left half of the tables, whereas the right half shows the MLP-CNN part which expands the supplemental scalar values onto two-dimensional data to match the data size with that of the target layer to be concatenated with. The last layer of the MLP-CNN part is then concatenated with the main stream network.

Table 1 Network structure of the CNN-MLP model used for force coefficient estimation of flow over a flat plate

Layer	Data size	Activation	Case	Layer	Data size	Activation	Case
Input (vorticity field)	(148,132,1)	–	All	Input ($\mathrm{Re_c}$, α)	(2)		1,2
Conv2D (7,7,16)	(148,132,32)	ReLU	All	Dense	(16)	ReLU	1,2
Concatenate	(148,132,34)	–	1	Dense	(32)	ReLU	1,2
Max Pooling	(74,66,32)	–	All	Dense	(64)	ReLU	1,2
Conv2D (7,7,16)	(74,66,32)	ReLU	All	Dense	(256)	ReLU	1,2
Max Pooling	(37,33,32)	–	All	Dense	(512)	ReLU	1,2
Conv2D (7,7,16)	(37,33,16)	ReLU	All	Dense	(1024)	ReLU	1,2
Concatenate	(37,33,18)	–	2	Dense	(1221)	ReLU	1,2
Conv2D (7,7,8)	(37,33,8)	ReLU	All	Reshape	(37,33,1)	–	1,2
Conv2D (7,7,4)	(37,33,4)	ReLU	All	Conv2D (7,7,2)	(37,33,2)	ReLU	1,2
Conv2D (7,7,1)	(37,33,1)	ReLU	All	UpSampling	(74,66,2)	–	1
Reshape	(1024)	–	All	Conv2D (7,7,2)	(74,66,2)	ReLU	1,2
Concatenate	(1026)	–	3	UpSampling	(148,132,2)	–	1
Dense	(1024)	ReLU	All	Conv2D (7,7,16)	(148,132,2)	ReLU	1,2
Dense	(256)	ReLU	All				
Dense	(64)	ReLU	All				
Concatenate	(66)	–	4				
Dense	(32)	ReLU	All				
Dense	(16)	ReLU	All				
Dense	(2)	Linear	All				

The left half of the table shows a mainstream of the network, while the right half shows the part which expands scalar values to fed into the mainstream network. For Conv2D layers, the size of filter H and the number of filters n are denoted as (H, H, n)

Table 2 Network structure of the CNN-MLP model used for force coefficient estimation of flow over two side-by-side cylinders

Layer	Data size	Activation	Case	Layer	Data size	Activation	Case
Input (vorticity field)	(240,448,1)	–	All	Input (g, r)	(2)	–	1,2
Conv2D (7,7,32)	(240,448,32)	ReLU	All	Dense	(16)	ReLU	1,2
Concatenate	(240,448,34)	–	1	Dense	(32)	ReLU	1,2
Max Pooling	(120,224,32)	–	All	Dense	(64)	ReLU	1,2
Conv2D (7,7,32)	(120,224,32)	ReLU	All	Dense	(256)	ReLU	1,2
Max Pooling	(60,112,32)	–	All	Dense	(420)	ReLU	1,2
Concatenate	(60,112,34)	–	2	Reshape	(15,28,1)	–	1,2
Conv2D (7,7,32)	(60,112,32)	ReLU	All	Conv2D (7,7,2)	(15,28,2)	ReLU	1,2
Max Pooling	(30,56,32)	–	All	UpSampling	(30,56,2)	–	1,2
Conv2D (7,7,32)	(30,56,32)	ReLU	All	Conv2D (7,7,2)	(30,56,2)	ReLU	1,2
Conv2D (7,7,16)	(30,56,16)	ReLU	All	UpSampling	(60,112,2)	–	1,2
Max Pooling	(15,28,16)	–	All	Conv2D (7,7,2)	(60,112,2)	ReLU	1,2
Conv2D (7,7,16)	(15,28,16)	ReLU	All	UpSampling	(120,224,2)	–	1
Conv2D (7,7,8)	(15,28,8)	ReLU	All	Conv2D (7,7,2)	(120,224,2)	ReLU	1,2
Conv2D (7,7,4)	(15,28,4)	ReLU	All	UpSampling	(240,448,2)	–	1
Reshape	(1680)	–	All	Conv2D (7,7,16)	(240,448,2)	ReLU	1,2
Concatenate	(1682)	–	3				
Dense	(1024)	ReLU	All				
Dense	(512)	ReLU	All				
Dense	(256)	ReLU	All				
Dense	(128)	ReLU	All				
Dense	(64)	ReLU	All				
Concatenate	(66)	–	4				
Dense	(32)	ReLU	All				
Dense	(16)	ReLU	All				
Dense	(4)	Linear	All				

The left half of the table shows a mainstream of the network, while the right half shows the part which expands scalar values to fed into the mainstream network. For Conv2D layers, the size of filter H and the number of filters n are denoted as (H, H, n)

Table 3 shows the structure of CNN-AE utilized in Sect. 3.2.4. The supplemental Reynolds numbers are expanded using the MLP-CNN part shown in the right half of the table. They are then concatenated with the mainstream network at four different placements.

Table 3 Network structure of the CNN-AE for two-dimensional decaying turbulence

Layer	Data size	Activation	Case	Layer	Data size	Activation	Case
Input (vorticity field)	(256,256,1)	–	All	Input (Re_0, Re_λ)	(2)	–	All
Conv2D (7,7,32)	(256,256,32)	ReLU	All	Dense	(16)	ReLU	All
Concatenate	(256,56,34)	–	1	Dense	(32)	ReLU	All
Max Pooling	(128,128,32)	–	All	Dense	(64)	ReLU	All
Conv2D (7,7,32)	(128,128,32)	ReLU	All	Dense	(256)	ReLU	1,2,4
Max Pooling	(64,64,32)	–	All	Reshape	(16,16,1)	ReLU	1,2,4
Conv2D (7,7,32)	(64,64,32)	ReLU	All	Conv2D (7,7,2)	(16,16,2)	ReLU	1,2,4
Max Pooling	(32,32,32)	–	All	UpSampling	(32,32,2)	–	1,2,4
Concatenate	(32,32,34)	ReLU	2	Conv2D (7,7,2)	(32,32,2)	ReLU	1,2,4
Conv2D (7,7,32)	(32,32,32)	ReLU	All	UpSampling	(64,64,2)	–	1
Max Pooling	(16,16,32)	–	All	Conv2D (7,7,2)	(64,64,2)	ReLU	1,2,4
Conv2D (7,7,16)	(16,16,32)	ReLU	All	UpSampling	(128,128,2)	–	1
Max Pooling	(8,8,32)	–	All	Conv2D (7,7,2)	(128,128,2)	ReLU	1,2,4
Conv2D (7,7,32)	(8,8,32)	ReLU	All	UpSampling	(255,256,2)	–	1
Conv2D (7,7,8)	(8,8,8)	ReLU	All	Conv2D (7,7,2)	(256,256,2)	ReLU	1,2,4
Conv2D (7,7,4)	(8,8,4)	ReLU	All	Conv2D (7,7,2)	(256,256,2)	ReLU	1,2,4
Conv2D (7,7,2)	(8,8,2)	ReLU	All				
Concatenate	(8,8,4)	ReLU	3				
Conv2D (7,7,2)	(8,8,2)	ReLU	All	Dense	(64)	ReLU	3
Conv2D (7,7,2)	(8,8,2)	ReLU	All	Reshape	(8,8,1)	ReLU	3
Conv2D (7,7,4)	(8,8,4)	ReLU	All	Conv2D (7,7,2)	(8,8,2)	ReLU	3
Conv2D (7,7,8)	(8,8,8)	ReLU	All	Conv2D (7,7,2)	(8,8,2)	ReLU	3
Conv2D (7,7,32)	(8,8,32)	ReLU	All				
UpSampling	(16,16,32)	–	All				
Conv2D (7,7,32)	(16,16,32)	ReLU	All				
UpSampling	(32,32,32)	–	All				
Concatenate	(32,32,34)	ReLU	4				
Conv2D (7,7,32)	(32,32,32)	ReLU	All				
UpSampling	(64,64,32)	–	All				
Conv2D (7,7,32)	(64,64,32)	ReLU	All				
UpSampling	(128,128,32)	–	All				
Conv2D (7,7,32)	(128,128,32)	ReLU	All				
UpSampling	(256,256,32)	–	All				
Conv2D (7,7,32)	(256,256,32)	ReLU	All				
Conv2D (7,7,1)	(256,256,1)	Linear	All				

The left half of the table shows a mainstream of the network, while the right half shows the part which expands scalar values to fed into the mainstream network. For Conv2D layers, the size of filter H and the number of filters n are denoted as (H, H, n)

Appendix B: Detailed observation for the robustness of the CNN-MLP model against noisy input

We provide the detailed information for the problem presented in Fig. 11, i.e., the robustness of the CNN-MLP model for noisy inputs. Figure 11 shows the ensemble L_2 error norm averaged over the coefficients of both cylinders for all considered distance factor g, in order to show the general trend of the model's response. In Figs. 18, 19 and 20, the result of each flow configurations and coefficients set is presented individually. From these results too, we can confirm that inserting scalar values at the earlier layer generally leads to the better solution.

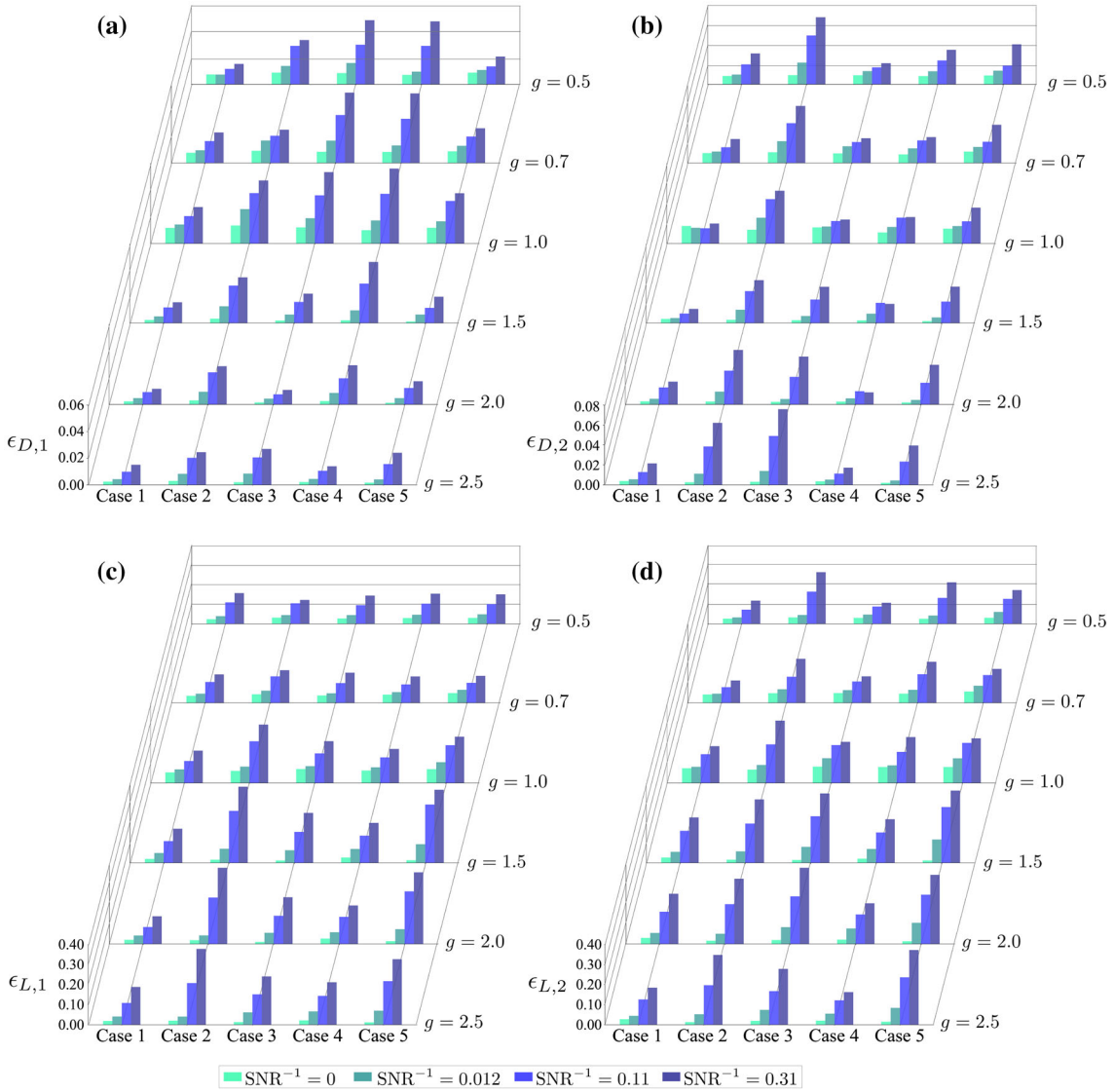

Fig. 18 Robustness against noisy input for a flow over two side-by-side cylinders with its radius ratio of $r = 1.00$

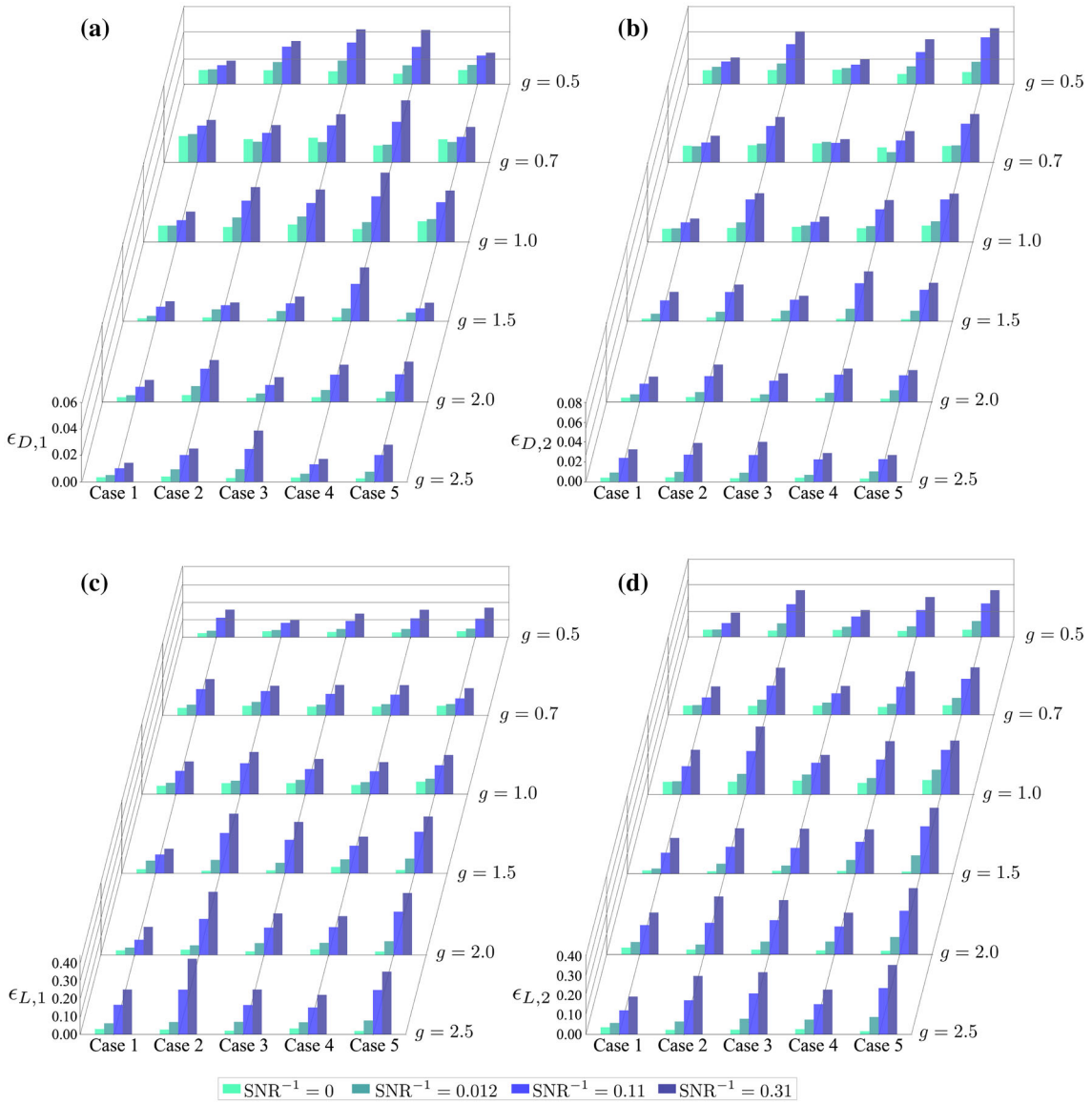

Fig. 19 Robustness against noisy input for a flow over two side-by-side cylinders with its radius ratio of $r = 1.15$

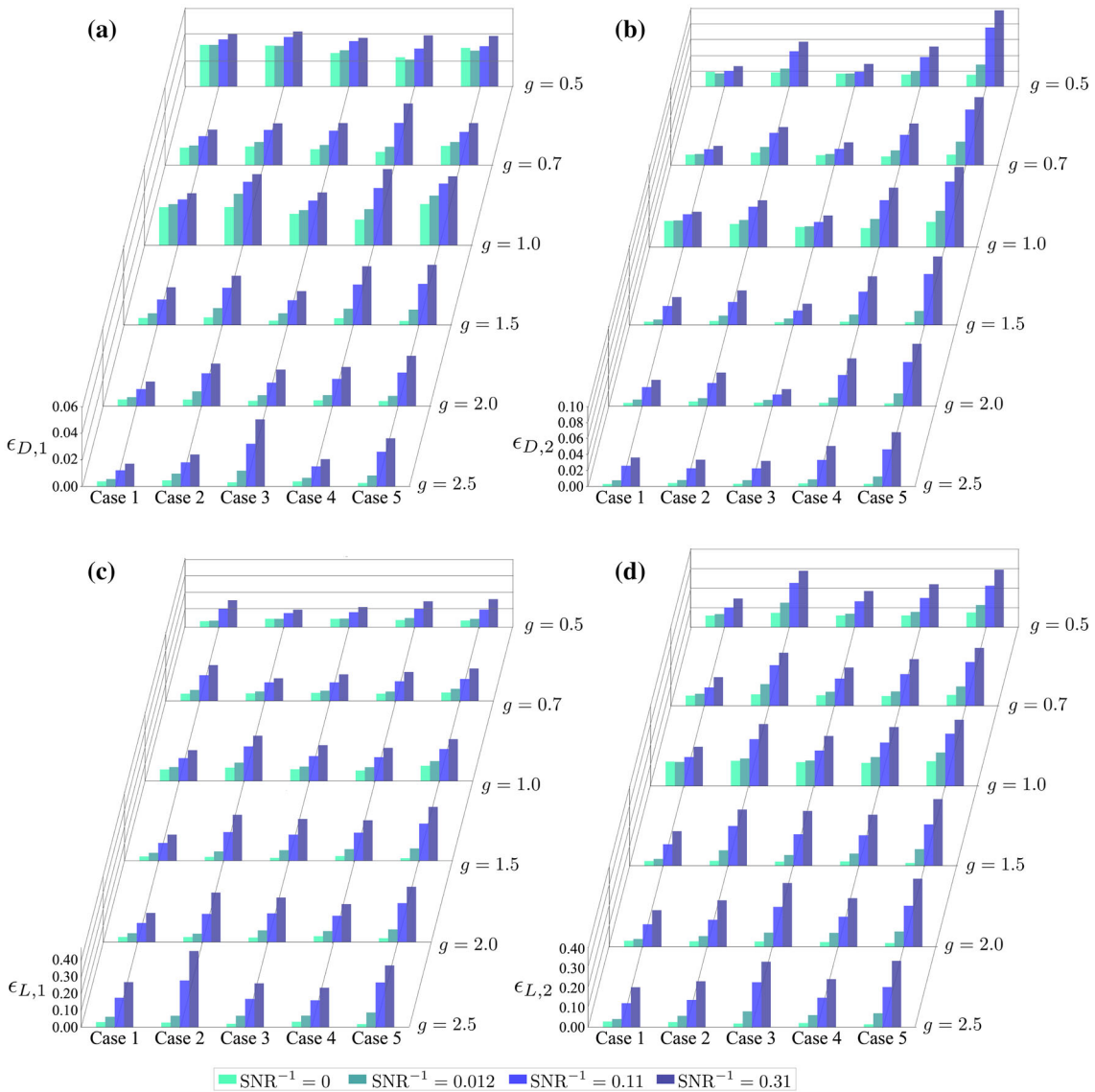

Fig. 20 Robustness against noisy input for a flow over two side-by-side cylinders with its radius ratio of $r = 1.30$

References

1. Fukami, K., Fukagata, K., Taira, K.: Assessment of supervised machine learning for fluid flows. Theor. Comput. Fluid Dyn. **34**(4), 497–519 (2020)
2. Duraisamy, K., Iaccarino, G., Xiao, H.: Turbulence modeling in the age of data. Annu. Rev. Fluid Mech. **51**, 357–377 (2019)
3. Brenner, M.P., Eldredge, J.D., Freund, J.B.: Perspective on machine learning for advancing fluid mechanics. Phys. Rev. Fluids **4**, 100501 (2019)
4. Nakamura, T., Fukami, K., Fukagata, K.: Comparison of linear regressions and neural networks for fluid flow problems assisted with error-curve analysis. arXiv:2105.00913 (2021)
5. Brunton, S.L., Hemati, M.S., Taira, K.: Special issue on machine learning and data-driven methods in fluid dynamics. Theor. Comput. Fluid Dyn. **34**, 333–337 (2020)
6. Duraisamy, K.: Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence. Phys. Rev. Fluids **6**, 050504 (2021)
7. Lapeyre, C.J., Misdariis, A., Cazard, N., Veynante, D., Poinsot, T.: Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates. Combust. Flame **203**, 255–264 (2019)
8. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagation errors. Nature **322**, 533–536 (1986)
9. Pawar, S., San, O., Rasheed, A., Vedula, P.: A priori analysis on deep learning of subgrid-scale parameterizations for Kraichnan turbulence. Theor. Comput. Fluid Dyn. **34**, 429–455 (2020)

10. Thuerey, N., Weißenow, K., Prantl, L., Hu, X.: Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows. AIAA J. **58**(1), 25–36 (2020)
11. Font, B., Weymouth, G.D., Nguyen, V.-T., Tutty, O.R.: Deep learning the spanwise-averaged Navier–Stokes equations. J. Comput. Phys. **434**, 110199 (2021)
12. Fukami, K., Fukagata, K., Taira, K.: Super-resolution reconstruction of turbulent flows with machine learning. J. Fluid Mech. **870**, 106–120 (2019)
13. Liu, B., Tang, J., Huang, H., Lu, X.-Y.: Deep learning methods for super-resolution reconstruction of turbulent flows. Phys. Fluids **32**, 025105 (2020)
14. Kim, J., Lee, C.: Deep unsupervised learning of turbulence for inflow generation at various Reynolds numbers. J. Comput. Phys. **406**, 109216 (2020)
15. Deng, Z., He, C., Liu, Y., Kim, K.C.: Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework. Phys. Fluids **31**, 125111 (2019)
16. Morimoto, M., Fukami, K., Fukagata, K.: Experimental velocity data estimation for imperfect particle images using machine learning. arXiv:2005.00756 (2020)
17. Cai, S., Zhou, S., Xu, C., Gao, Q.: Dense motion estimation of particle images via a convolutional neural network. Exp. Fluids **60**, 60–73 (2019)
18. Salehipour, H., Peltier, W.R.: Deep learning of mixing by two 'atoms' of stratified turbulence. J. Fluid Mech. **861**, R4 (2019)
19. Morimoto, M., Fukami, K., Zhang, K., Fukagata, K.: Generalization techniques of neural networks for fluid flow estimation. arXiv:2011.11911 (2020)
20. Murata, T., Fukami, K., Fukagata, K.: Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. J. Fluid Mech. **882**, A13 (2020)
21. Milano, M., Koumoutsakos, P.: Neural network modeling for near wall turbulent flow. J. Comput. Phys. **182**, 1–26 (2002)
22. Lusch, B., Kutz, J.N., Brunton, S.L.: Deep learning for universal linear embeddings of nonlinear dynamics. Nat. Commun. **9**, 4956 (2018)
23. Fukami, K., Nakamura, T., Fukagata, K.: Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. Phys. Fluids **32**, 095110 (2020)
24. Hasegawa, K., Fukami, K., Murata, T., Fukagata, K.: Data-driven reduced order modeling of flows around two-dimensional bluff bodies of various shapes. In: ASME-JSME-KSME Joint Fluids Engineering Conference, San Francisco, USA (Paper 5079) (2019)
25. Maulik, R., Lusch, B., Balaprakash, P.: Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. Phys. Fluids **33**, 037106 (2021)
26. Fukami, K., Murata, T., Fukagata, K.: Sparse identification of nonlinear dynamics with low-dimensionalized flow representations. arXiv:2010.12177 (2020)
27. Carlberg, K.T., Jameson, A., Kochenderfer, M.J., Morton, J., Peng, L., Witherden, F.D.: Recovering missing CFD data for high-order discretizations using deep neural networks and dynamics learning. J. Comput. Phys. **395**, 105–124 (2019)
28. Fukami, K., Nabae, Y., Kawai, K., Fukagata, K.: Synthetic turbulent inflow generator using machine learning. Phys. Rev. Fluids **4**, 064603 (2019)
29. Nakamura, T., Fukami, K., Hasegawa, K., Nabae, Y., Fukagata, K.: Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. Phys. Fluids **33**, 025116 (2021)
30. Liu, Y., Ponce, C., Brunton, S.L., Kutz, J.N.: Multiresolution convolutional autoencoders. arXiv:2004.04946 (2020)
31. Lee, S., You, D.: Data-driven prediction of unsteady flow fields over a circular cylinder using deep learning. J. Fluid Mech. **879**, 217–254 (2019)
32. Ling, J., Kurzawski, A., Templeton, J.: Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J. Fluid Mech. **807**, 155–166 (2016)
33. Hasegawa, K., Fukami, K., Murata, T., Fukagata, K.: Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. Theor. Comput. Fluid Dyn. **34**(4), 367–388 (2020)
34. Hasegawa, K., Fukami, K., Murata, T., Fukagata, K.: CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers. Fluid Dyn. Res. **52**(6), 065501 (2020)
35. Zhang, Y., Sung, W.J., Mavris, D.N.: Application of convolutional neural network to predict airfoil lift coefficient. In: AIAA paper, pp. 2018–1903 (2018)
36. Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., Kaushik, S.: Prediction of aerodynamic flow fields using convolutional neural networks. Comput. Mech. **64**(2), 525–545 (2019)
37. Xu, J., Duraisamy, K.: Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. Comput. Methods Appl. Mech. Eng. **372**, 113379 (2020)
38. Zafar, M.I., Xiao, H., Choudhari, M.M., Li, F., Chang, C.-L., Paredes, P., Venkatachari, B.: Convolutional neural network for transition modeling based on linear stability theory. Phys. Rev. Fluids **5**, 113903 (2020)
39. Pawar, S., San, O., Aksoylu, B., Rasheed, A., Kvamsdal, T.: Physics guided machine learning using simplified theories. Phys. Fluids **33**, 011701 (2021)
40. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
42. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
43. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
44. Kim, J., Lee, C.: Prediction of turbulent heat transfer using convolutional neural networks. J. Fluid Mech. **882**, A18 (2020)
45. Huang, J., Liu, H., Cai, W.: Online in situ prediction of 3-D flame evolution from its history 2-D projections via deep learning. J. Fluid Mech. **875**, R2 (2019)

46. Gakhar, S., Koseff, J.R., Ouellette, N.T.: On the surface expression of bottom features in free-surface flow. J. Fluid Mech. **900**, A41 (2020)
47. Matsuo, M., Nakamura, T., Morimoto, M., Fukami, K., Fukagata, K.: Supervised convolutional network for three-dimensional fluid data reconstruction from sectional flow fields with adaptive super-resolution assistance. arXiv:2103.09020 (2021)
48. Moriya, N., Fukami, K., Nabae, Y., Morimoto, M., Nakamura, T., Fukagata, K.: Inserting machine-learned virtual wall velocity for large-eddy simulation of turbulent channel flows. arXiv:2106.09271 (2021)
49. Fukami, K., Fukagata, K., Taira, K.: Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. J. Fluid Mech. **909**, A9 (2021)
50. Fukami, K., Hasegawa, K., Nakamura, T., Morimoto, M., Fukagata, K.: Model order reduction with neural networks: application to laminar and turbulent flows. arXiv:2011.10277 (2020)
51. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: In Proceedings of 27th International Conference on Machine Learning (2010)
52. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
53. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
54. Peng, J., Liu, X., Aubry, N., Chen, Z., Wu, W.T.: Data-driven modeling of geometry-adaptive steady heat transfer based on convolutional neural networks: heat conduction. arxiv:2010.03854 (2020)
55. Dosovitskiy, A., Fischer, P., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with exemplar convolutional neural networks. IEEE Trans. Pattern Anal. Mach. Intell. **38**, 1734–1747 (2019)
56. Dumoulin, V., Visin, F.: A guide to convolution arithmetic for deep learning. arXiv:1603.07285 (2016)
57. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015)
58. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
59. Omata, N., Shirayama, S.: A novel method of low-dimensional representation for temporal behavior of flow fields using deep autoencoder. AIP Adv. **9**(1), 015006 (2019)
60. Brunton, S.L., Kutz, J.N.: Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control. Cambridge University Press, Cambridge (2019)
61. Taira, K., Colonius, T.: The immersed boundary method: a projection approach. J. Comput. Phys. **225**(2), 2118–2137 (2007)
62. Colonius, T., Taira, K.: A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. Comput. Methods Appl. Mech. Eng. **197**, 2131–2146 (2008)
63. Weller, H.G., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput. Phys. **12**(6), 620–631 (1998)
64. Taira, K., Nair, A.G., Brunton, S.L.: Network structure of two-dimensional decaying isotropic turbulence. J. Fluid Mech. **795**, R2 (2016)
65. Shanker, M., Hu, M.Y., Hung, M.S.: Effect of data standardization on neural network training. Omega **24**(4), 385–397 (1996)
66. Domingos, P.: A few useful things to know about machine learning. Commun. ACM **55**(10), 78–87 (2012)
67. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of Computer Vision and Pattern Recognition, pp. 770–778 (2016)
68. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. In: Distill (2016)
69. Ogoke, F., Meidani, K., Hashemi, A., Farimani, A.B.: Graph convolutional neural networks for body force prediction. arXiv:2012.02232 (2020)
70. Tencer, J., Potter, K.: Enabling nonlinear manifold projection reduced-order models by extending convolutional neural networks to unstructured data. arXiv:2006.06154 (2020)
71. Kashefi, A., Rempe, D., Guibas, L.J.: A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. Phys. Fluids **33**, 027104 (2021)
72. Gao, H., Sun, L., Wang, J.-X.: PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. J. Comput. Phys. **428**, 110079 (2021)
73. Fukami, K., Maulik, R., Ramachandra, N., Fukagata, K., Taira, K.: Global field reconstruction from sparse sensors with Voronoi tessellation-assisted deep learning. arXiv:2101.00554 (2021)
74. Maulik, R., Fukami, K., Ramachandra, N., Fukagata, K., Taira, K.: Probabilistic neural networks for fluid flow surrogate modeling and data recovery. Phys. Rev. Fluids **5**, 104401 (2020)
75. Erichson, N.B., Mathelin, L., Yao, Z., Brunton, S.L., Mahoney, M.W., Kutz, J.N.: Shallow neural networks for fluid flow reconstruction with limited sensors. Proc. Royal Soc. A **476**(2238), 20200097 (2020)