Data-driven prediction of unsteady flow fields over a circular cylinder using deep learning

Article · April 2018

CITATIONS
6

READS
74

2 authors, including:

Sangseung Lee
Pohang University of Science and Technology
8 PUBLICATIONS **34** CITATIONS

SEE PROFILE

Data-driven prediction of unsteady flow over a circular cylinder using deep learning

Sangseung Lee and Donghyun You†

Department of Mechanical Engineering, Pohang University of Science and Technology, 77 Cheongam-ro, Nam-gu, Pohang, Gyeongbuk 37673, Republic of Korea

(Received xx; revised xx; accepted xx)

Unsteady flow fields over a circular cylinder are trained and predicted using four different deep learning networks: convolutional neural networks with and without consideration of conservation laws, generative adversarial networks with and without consideration of conservation laws. Flow fields at future occasions are predicted based on information of flow fields at previous occasions. Deep learning networks are trained first using flow fields at Reynolds numbers of 100, 200, 300, and 400, while flow fields at Reynolds numbers of 500 and 3000 are predicted using the trained deep learning networks. Physical loss functions are proposed to explicitly impose information of conservation of mass and momentum to deep learning networks. An adversarial training is applied to extract features of flow dynamics in an unsupervised manner. Effects of the proposed physical loss functions, adversarial training, and network sizes on the prediction accuracy are analyzed. Predicted flow fields using deep learning networks are in favorable agreement with flow fields computed by numerical simulations.

Key words: To be added during the typesetting process

1. Introduction

Observation of fluid flow in nature, laboratory experiments, and numerical simulations has provided evidence of existence of flow features and certain, but often complex, ordinance. For example, in nature, Kelvin-Helmholtz waves in the cloud (Dalin *et al.* 2010), von Karman vortices in ocean flow around an island (Berger & Wille 1972), and swirling great red spot on the Jupiter (Marcus 1988) are flow structures that can be classified as a certain type of vortical motions produced by distinct combination of boundary conditions and initial conditions for the governing first principles. Similar observation also has been reported in laboratory experiments and numerical simulations (Freymuth 1966; Ruderich & Fernholz 1986; Wu & Moin 2009; Babucke *et al.* 2008). The existence of distinct and dominant flow features that are linearly independent have also been widely investigated by mathematical decomposition techniques such as the proper orthogonal decomposition (POD) method (Sirovich 1987), the dynamic mode decomposition (DMD) method (Schmid 2010), and the Koopman operator method (Mezić 2013).

For the sake of the existence of distinct or dominant flow features, animals such as insects, birds, and fish are reported to control their bodies adequately to better adapt the fluid dynamic environment and to improve the aero- or hydro-dynamic performance and efficiency (Wu 2011; Yonehara *et al.* 2016). This suggests a possibility that they

† Email address for correspondence: dhyou@postech.ac.kr

empirically learn dominant fluid motions as well as the non-linear correlation of fluid motions and are able to estimate future flow based on experienced flow in their living environments. Such observation in nature motivates us to investigate the feasibility of predicting unsteady fluid motions by learning flow features using neural networks.

Attempts to apply neural networks to problems of fluid flow have been recently conducted by Tracey *et al.* (2015), Zhang & Duraisamy (2015), and Singh *et al.* (2017), who utilized shallow neural networks for turbulence modeling for Reynolds-averaged Navier-Stokes (RANS) simulations. Ling *et al.* (2016) employed deep neural networks to better model the Reynolds stress anisotropy tensor for RANS simulations. Guo *et al.* (2016) employed a convolutional neural network (CNN) to predict steady flow fields around bluff objects and reported reasonable prediction of steady flow fields with significantly reduced computational cost than that required for numerical simulations. Similarly, Miyanawala & Jaiman (2017) employed a CNN to predict aerodynamic force coefficients of bluff bodies, also with notably reduced computational costs. Those previous studies showed high potential of deep learning techniques for enhancing simulation accuracy and reducing computational cost.

Predicting unsteady flow fields using deep learning involves extracting both spatial and temporal features of input flow field data, which could be considered as learning videos. Video modeling enables prediction of a future frame of a video based on information of previous video frames by learning spatial and temporal features of the video. Although deep learning techniques have been reported to generate high quality real-world like images in image modeling areas (Radford *et al.* 2015; Denton *et al.* 2015; van den Oord *et al.* 2016*b*,*a*), it is known that, for video modeling, deep learning techniques have shown difficulties in generating high quality prediction due to blurriness caused by complexity in the spatial and temporal features in a video (Srivastava *et al.* 2015; Ranzato *et al.* 2014; Mathieu *et al.* 2015).

Mathieu *et al.* (2015) proposed a video modeling architecture that utilizes a generative adversarial network (GAN) with multi-scale CNNs to generate prediction of future frames, by combining generator and discriminator models (Goodfellow *et al.* 2014). The generator model generates images and the discriminator model is employed to discriminate the generated images from real (ground truth) images. A GAN is adversarially trained so the generator network is trained to delude the discriminator network, and the discriminator network is trained not to be deluded by the generator network. The Nash equilibrium inherently presence in the two pronged adversarial training and it leads the network to extract underlying low-dimensional features in an unsupervised manner, in consequence, better quality images can be generated. Mathieu *et al.* (2015) reported that prediction of future frames of a video has improved compared to a method based on CNNs and recurrent neural networks (RNN) (Ranzato *et al.* 2014), by utilizing a GAN with multi-scale CNNs and a gradient difference loss (GDL) function.

Prediction of unsteady flow fields using deep learning could offer new opportunities for real-time control and guidance of aero- or hydro-vehicles, fast weather forecast, *etc.* As the first step towards prediction of unsteady flow fields using deep learning, in the present study, it is attempted to predict rather simple but canonical unsteady vortex shedding over a circular cylinder using four different deep learning networks: CNNs with and without consideration of conservation laws, GANs with and without consideration of conservation laws. Consideration of conservation laws is realized as a form of loss functions.

The paper is organized as follows. The method for constructing flow field datasets is explained in section 2. The present deep learning method and results obtained using deep

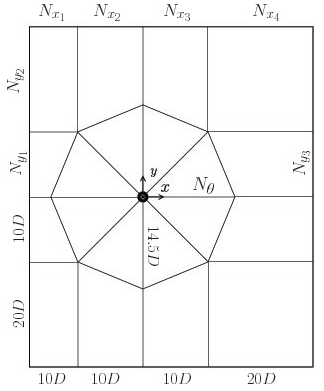


Figure 1: The computational domain for numerical simulations. N denotes the number of mesh points, where $N_{x_1} = 20$, $N_{x_2} = 30$, $N_{x_3} = 50$, $N_{x_4} = 50$ $N_{y_1} = 30$, $N_{y_2} = 30$, $N_{y_3} = 80$, and $N_\theta = 150$. The domain size and number of mesh points in the spanwise direction are $6D$ and 96, respectively.

learning methods are presented in sections 3 and 4, respectively, followed by concluding remarks in section 5.

2. Construction of flow field datasets

2.1. *Numerical simulations*

Numerical simulations of flow over a circular cylinder at $Re_D = U_\infty D/\nu = 100, 200, 300, 400, 500,$ and 3000, where U_∞, D, and ν are the freestream velocity, cylinder diameter, and kinematic viscosity, respectively, have been conducted by solving the incompressible Navier-Stokes equations as follows:

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial u_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{2.1}$$

and

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2.2}$$

where u_i, p, and ρ are the velocity, pressure, and density, respectively. Velocity components and the pressure are non-dimensionalized by U_∞ and ρU_∞^2, respectively. A fully implicit fractional step method is employed for time integration, where all terms in the Navier-Stokes equations are integrated using the Crank-Nicolson method. Second-order central-difference schemes are employed for spatial discretization (You *et al.* 2008). The computational domain consists of a block structured H-grid with an O-grid around the cylinder (figure 1). The computational domain sizes are $50D$, $60D$, and $6D$ in the streamwise, cross-flow, and spanwise directions, respectively, where D is the cylinder diameter. The computational time-step size ($\Delta t U_\infty/D$) of 0.005 is used for all simulations. The domain size, number of grid points, and time-step sizes are determined from an extensive sensitivity study.

2.2. *Datasets*

Simulation results of flow over a cylinder at each Reynolds number are collected with a time-step interval of $\delta t = 20\Delta t U_\infty/D = 0.01$. Flow variables ($u_1/U_\infty (= u/U_\infty)$,

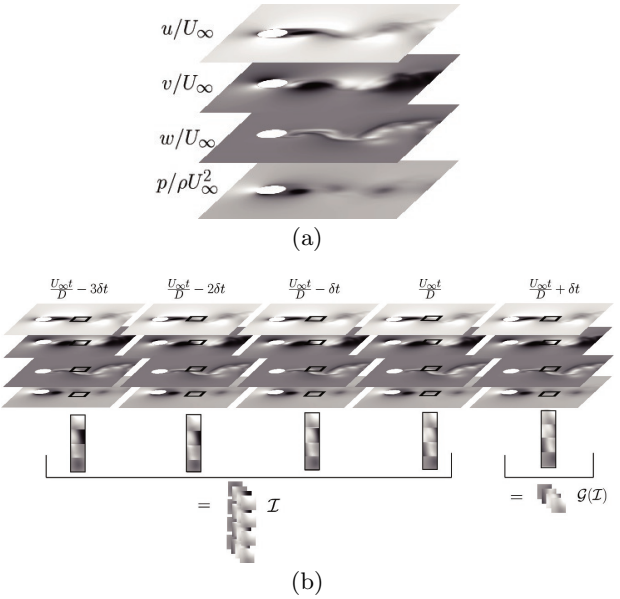


Figure 2: (a) Instantaneous fields of flow variables $u/U_\infty, v/U_\infty, w/U_\infty$, and $p/\rho U_\infty^2$ on a $7D \times 7D$ domain with 250×250 grid cells. (b) The procedure of subsampling five consecutive flow fields to the input ($\mathcal{I}$) and the ground truth ($\mathcal{G}(\mathcal{I})$) on a $0.896D \times 0.896D$ domain with 32×32 grid cells.

$u_2/U_\infty (= v/U_\infty)$, $u_3/U_\infty (= w/U_\infty)$, and $p/\rho U_\infty^2$) at each time-step in a square domain of $-1.5D < x < 5.5D$, $-3.5D < y < 3.5D$, $z = 0D$ ($7D \times 7D$ sized domain) are extracted into an uniform grid with the size of 250×250 grid cells for all Reynolds number cases. Thus, a dataset at each Reynolds number consists of flow fields with the size of 250×250 (grid cells) $\times$ 4 (flow variables).

Datasets of flow fields at $Re = 100, 200, 300$, and 400 are employed for training deep learning networks. Flow fields in the training dataset are randomly subsampled in time and space to five consecutive flow fields on a 32×32 sized grid and $0.896D \times 0.896D$ sized domain (see figure 2). The subsampled flow fields contain diverse types of flow such as, freestream flow, wake flow, boundary layer flow, or separating flow. Therefore, deep learning networks are allowed to learn diverse types of flow. The first four consecutive sets of flow fields are used as an input ($\mathcal{I}$), while the following set of flow fields is a ground truth flow field ($\mathcal{G}(\mathcal{I})$). The pair of input and ground truth flow fields form a training sample. In the present study, total 500,000 training samples are employed for training deep learning networks. The predictive performances of networks are evaluated on a test dataset, which is composed of flow fields at $Re = 500$ and 3000 on the 250×250 sized grid in the $7D \times 7D$ sized domain, without any subsampling in space.

3. Deep learning methodology

3.1. *Overall procedure of deep learning*

A deep learning network learns a nonlinear mapping of an input tensor and an output tensor. The nonlinear mapping is comprised of a sequence of tensor operations and nonlinear activations of weight parameters. The objective of deep learning is to learn appropriate weight parameters that forms the most accurate nonlinear mapping of the

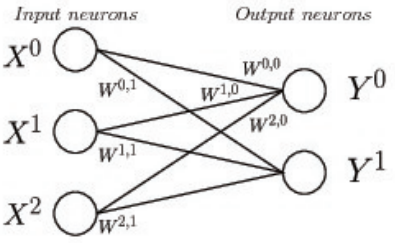


Figure 3: Illustration of a fully connected layer.

input tensor and the output tensor that minimizes a loss function. A loss function evaluates the difference between the estimated output tensor and the ground truth output tensor (the desired output tensor). Therefore, deep learning is an optimization procedure for determining weight parameters that minimize a loss function. A deep learning network is trained with the following steps:
1. A network estimates an output tensor from a given input through the current state of weight parameters, which is known as feed forward.
2. A loss (scalar value) is evaluated by a loss function of the difference between the estimated output tensor and the ground truth output tensor.
3. Gradients of the loss respect to each weight parameter are calculated through the chain rule of partial derivatives starting from the output tensor, which is known as back propagation.
4. The weight parameters are gradually updated in the negative direction of the gradients of the loss respect to each weight parameter.
5. Step 1 to 4 are repeated until weight parameters (deep learning network) are sufficiently updated.

The present study utilizes two different layers that contains weight parameters: fully connected layers and convolution layers. An illustration of a fully connected layer is shown in figure 3. Weight parameters of a fully connected layer are stored in connections (W) between layers of input (X) and output (Y) neurons, where neurons are elementary units in a fully connected layer. Information inside input neurons are passed to output neurons through a matrix multiplication of the weight parameter matrix and the vector of input neurons as follows:

$$Y^i = \sum_j W^{j,i} X^j + bias, \tag{3.1}$$

where a *bias* is a constant, which is also a parameter to be learned. An output neuron of a fully connected layer collects information from all input neurons with respective weight parameters. This provides a strength to learn a complex mapping of input and output neurons. However, as the number of weight parameters is determined as the multiplication of the number of input and output neurons, which number of neurons is generally in the order of hundreds or thousands, the number of weight parameters easily becomes more than sufficient. As a result, abundant use of fully connected layers leads to inefficient learning. Because of the reason, fully connected layers are typically used as a classifier, which collects information and classifies labels, after extracting features using convolution layers.

An illustration of a convolution layer is shown in figure 4. Weight parameters of a convolution layer are stored in kernels (W) between input (X) and output (Y) feature maps, where feature maps are elementary units in a convolution layer. To maintain the

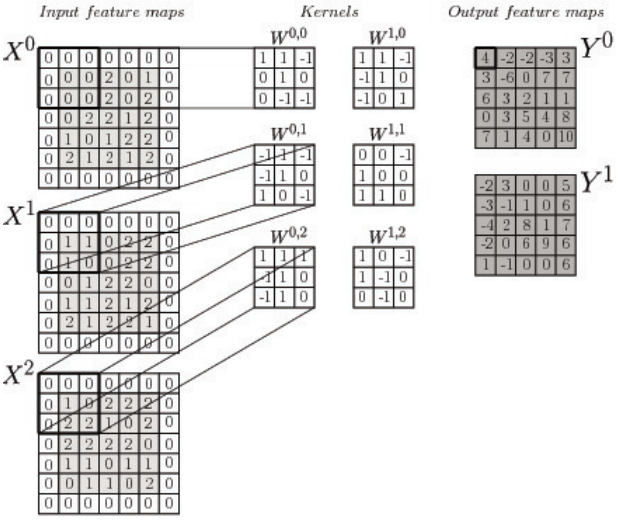


Figure 4: Illustration of a convolution layer.

shape of the input after convolution operations, zeros are padded around input feature maps. The convolution operation is applied to input feature maps using kernels as follows:

$$Y^n_{i,j} = \left[\sum_k \sum_{c=0}^{F_y-1} \sum_{r=0}^{F_x-1} W^{n,k}_{r,c} \underbrace{X^k_{i+r,j+c}}_{Pad\ included} \right] + bias, \tag{3.2}$$

where $F_x \times F_y$ is the size of kernels. Weight parameters inside kernels are updated to extract important spatial features inside input feature maps, so an output feature map contains a refined feature from input feature maps. Convolution layers contain significantly less amount of parameters to update, compared to fully connected layers, which enables efficient learning. Therefore, convolution layers are typically used for feature extraction.

After each fully connected layer or convolution layer, a nonlinear activation function is applied (but not always) to the output neurons or feature maps to provide nonlinearity to a deep learning network. Hyperbolic tangent function ($f(x) = \tanh(x)$), sigmoid function ($f(x) = 1/(1 + \exp^{-x})$), or a rectified linear unit (ReLU) activation function ($f(x) = \max(0, x)$) (Krizhevsky *et al.* 2012) are examples of typically applied activation functions.

3.2. *Configurations of deep learning networks*

Deep learning networks employed in the present study for flow prediction are variations of the architecture for video modeling proposed by Mathieu *et al.* (2015). The network structure in the present study consists of a generator model that accepts four consecutive sets of flow fields as an input. Each input set of flow fields is composed of flow variables of $\{u/U_\infty, v/U_\infty, w/U_\infty, p/\rho U^2_\infty\}$, to take an advantage of learning correlated physical phenomena between each flow variable. The generator model utilized in this study is composed of a set of multi-scale generative CNNs $\{G_0, G_1, G_2, G_3\}$ to learn multi-range spatial dependencies of flow structures (see table 1 and figure 5).

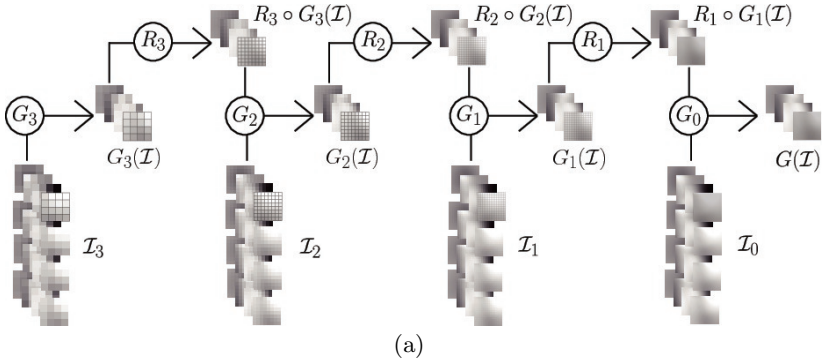
During training, a generative CNN G_k generates flow field predictions ($G_k(\mathcal{I})$) on a $\frac{32}{2^k} \times \frac{32}{2^k}$ sized grid through padded convolution layers. G_k is fed with four consecutive

GM_{16}		
Generative network	Feature map sizes	Kernel sizes
G_3	16 $\mathcal{N}_1$ $\mathcal{N}_1$ 4	3×3, 3×3, 3×3
G_2	20 $\mathcal{N}_1$ $\mathcal{N}_1$ 4	5×5, 3×3, 5×5
G_1	20 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	5×5, 3×3, 3×3, 3×3, 5×5
G_0	20 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	7×7, 5×5, 5×5, 5×5, 7×7
GM_{18}		
Generative network	Feature map sizes	Kernel sizes
G_3	16 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	3×3, 3×3, 3×3, 3×3
G_2	20 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	5×5, 3×3, 3×3, 5×5
G_1	20 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	5×5, 3×3, 3×3, 3×3, 5×5
G_0	20 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	7×7, 5×5, 5×5, 5×5, 7×7
GM_{20}		
Generative network	Feature map sizes	Kernel sizes
G_3	16 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	3×3, 3×3, 3×3, 3×3
G_2	20 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	5×5, 3×3, 3×3, 5×5
G_1	20 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_3$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	5×5, 3×3, 3×3, 3×3, 3×3, 5×5
G_0	20 $\mathcal{N}_1$ $\mathcal{N}_2$ $\mathcal{N}_3$ $\mathcal{N}_2$ $\mathcal{N}_1$ 4	7×7, 5×5, 5×5, 5×5, 5×5, 7×7
Number sets		
N_{32}	N_{64}	N_{128}
$\mathcal{N}_1 = 32$, $\mathcal{N}_2 = 64$, $\mathcal{N}_3 = 128$,	$\mathcal{N}_1 = 64$, $\mathcal{N}_2 = 128$, $\mathcal{N}_3 = 256$	$\mathcal{N}_1 = 128$, $\mathcal{N}_2 = 256$, $\mathcal{N}_3 = 512$

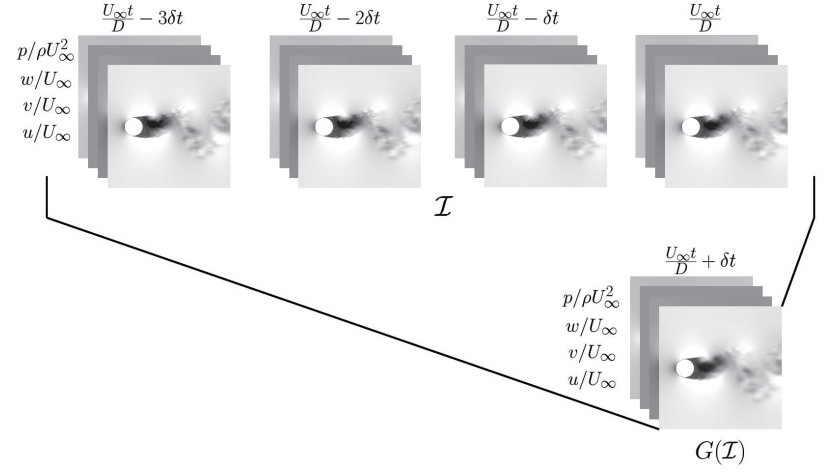
Table 1: Configurations (GM_{16}, GM_{18}, and GM_{20}) and number sets (N_{32}, N_{64}, and N_{128}) of generator models.

sets of flow fields on a $\frac{32}{2^k} \times \frac{32}{2^k}$ sized grid ($\mathcal{I}_k$), which are bilinearly interpolated from the original 32×32 sized input sets of flow fields ($\mathcal{I}$), and a set of upscaled flow fields, which is obtained by $R_{k+1} \circ G_{k+1}(\mathcal{I})$ (see figure 5). $R_{k+1} \circ ()$ is an upscale operator that bilinearly interpolates a flow field on a grid with the size of $\frac{32}{2^{k+1}} \times \frac{32}{2^{k+1}}$ to a grid with the size of $\frac{32}{2^k} \times \frac{32}{2^k}$. Note that domain sizes of the $\frac{32}{2^k} \times \frac{32}{2^k}$ and 32×32 grids are identical, where the corresponding convolution kernel size ranges from 3 to 7 (see table 1). Consequently, G_k is possible to learn larger spatial dependencies of flow fields than G_{k-1} by sacrificing resolution. As a result, a multi-scale CNN-based generator model enables to learn and predict flow fields with multiscale flow phenomena. Generator models with different sets of numbers of feature map sizes (N_{32}, N_{64}, and N_{128}) and numbers of weight layers (GM_{16} ,GM_{18}, and GM_{20}) are trained (see table 1). The last layer of feature maps in each multi-scale CNN is activated with the hyperbolic tangent function, while other feature maps are activated with the ReLU activation function.

Let $\mathcal{G}_k(\mathcal{I})$ be ground truth flow fields resized on a $\frac{32}{2^k} \times \frac{32}{2^k}$ grid. The discriminator model consists of a set of discriminative networks $\{D_0, D_1, D_2, D_3\}$ with convolution layers and fully connected layers (see table 2 and figure 6). A discriminative network,



(a)



(b)

Figure 5: (a) Schematic diagram of generator models. $\mathcal{I}$ is the set of input flow fields (see figure 2) and $\mathcal{I}_k$ denotes interpolated input flow fields on an identical domain with $1/(2^k \times 2^k)$ coarser grid resolution. G_k indicates a generative CNN which is fed with input I_k, while $G_k(\mathcal{I})$ indicates the set of predicted flow fields from the generative CNN. G_k. $R_k\circ$ indicates the rescale operator, which upscales the grid size twice in both directions. (b) Example of input flow fields and the corresponding flow field predictions on a test data.

D_k, is fed with inputs of predicted flow fields from the generative CNN ($G_k(\mathcal{I})$) and ground truth flow fields ($\mathcal{G}_k(\mathcal{I})$). Convolution layers of a discriminative network extracts low-dimensional features or representations of predicted flow fields and ground truth flow fields through convolution operations. 2×2 max pooling, which is an operation that extracts the maximum value from each equally divided 2×2 sized grid on a feature map, is added after convolution layers to pool the most important features. Then, fully connected layers compare pooled features to classify ground truth flow fields into class 1 and predicted flow fields into class 0. The output of each discriminative network is a single continuous scalar between 0 and 1, where an output value larger than a threshold (0.5) is classified into class 1 and an output value smaller than the threshold is classified into class 0. Output neurons of the last fully connected layer of each discriminative network D_k is activated using the sigmoid function, while other output neurons, including feature

Discriminator model		
Convolution layer		
Discriminative network	Feature map sizes	Kernel sizes
D_3	4 64	3×3
D_2	4 64 128 128	3×3, 3×3, 3×3
D_1	4 128 256 256	5×5, 5×5, 5×5
D_0	4 128 256 512 128	7×7, 7×7, 5×5, 5×5
2×2 max pooling		
Fully connected layer		
Discriminative network	Neuron numbers	
D_3	512 256 1	
D_2	1024 512 1	
D_1	1024 512 1	
D_0	1024 512 1	

Table 2: Configuration of the discriminator model.

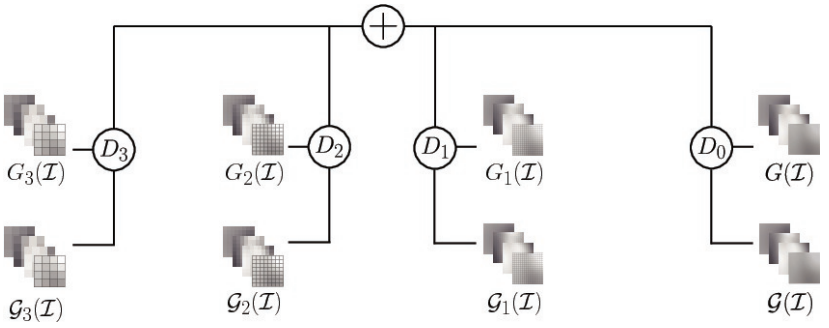


Figure 6: Schematic diagram of the discriminator model. D_k indicates the discriminative network which is fed with $G_k(\mathcal{I})$ and $\mathcal{G}_k(\mathcal{I})$. $G_k(\mathcal{I})$ indicates the set of predicted flow fields from the generative CNN G_k, while $\mathcal{G}_k(\mathcal{I})$ indicates the set of ground truth flow fields.

maps of convolution layers, are activated with the ReLU activation function. Generator models are trained with an Adam optimizer (Kingma & Ba 2014) with a learning rate, which determines the network update speed, of 4×10^{-5} and the discriminator model is trained with the gradient descent method with the learning rate of 0.02. Networks are trained up to 6×10^5 iterations with a batch size of 8, and the network with the best performance on predicting flow fields in test data during iterations is employed for qualitative and quantitative analyses.

3.3. *Conservation principles*

Let Ω be an arbitrary open, bounded, and connected domain in $\mathbb{R}^3$, $\partial\Omega$ be a surface of which an outward unit normal vector can be defined as $\hat{n} = (n^1, n^2, n^3)$. Also let $\rho(t, \vec{x})$

be the density, $\vec{u}(t,\vec{x}) = (u_1, u_2, u_3)$ be a velocity vector, $p(t,\vec{x})$ be the pressure, and $\tau(t,\vec{x})$ be a shear stress tensor ($\tau_{ij} = \rho\nu\frac{\partial u_j}{\partial x_i}$) of ground truth flow fields as a function of time t and space $\vec{x} \in \mathbb{R}^3$. Then conservation laws for mass and momentum can be written as follows:

$$\frac{d}{dt}\int_\Omega \rho dV = -\int_{\partial\Omega} \rho u_j n^j dS, \tag{3.3}$$

and

$$\frac{d}{dt}\int_\Omega \rho u_i dV = -\int_{\partial\Omega} (\rho u_i) u_j n^j dS - \int_{\partial\Omega} (p\delta_{ij}) n^j dS + \int_{\partial\Omega} \tau_{ji} n^j dS, \tag{3.4}$$

where δ_{ij} is the Kronecker delta. In the present study, conservation principles of mass and momentum in a flow field predicted by deep learning are written in a form that compares the difference between predicted and ground truth flow fields as follows:

$$\epsilon_c = \sum_{i=1}^{3} \left| \int_{\partial\Omega} u_i n^i - \widetilde{u}_i n^i dS \right|, \tag{3.5}$$

and

$$\begin{aligned}\epsilon_{mom} = \sum_{i=1}^{3}\sum_{j=1}^{3} \left| \int_{\partial\Omega} u_i u_j n^j - \widetilde{u}_i \widetilde{u}_j n^j dS \right| + \\ \left| \int_{\partial\Omega} \frac{p\delta_{ij}}{\rho} n^j - \frac{\widetilde{p}\delta_{ij}}{\rho} n^j dS \right| + \\ \left| \int_{\partial\Omega} \frac{\widetilde{\tau}_{ji}}{\rho} n^j - \frac{\tau_{ji}}{\rho} n^j dS \right|,\end{aligned} \tag{3.6}$$

where $\vec{\widetilde{u}}(t,\vec{x}) = (\widetilde{u}_1, \widetilde{u}_2, \widetilde{u}_3)$ is the velocity vector, $\widetilde{p}(t,\vec{x})$ is the pressure, and $\widetilde{\tau}(t,\vec{x})$ is the shear stress tensor of predicted flow fields in functions of time t and space $\vec{x} \in \mathbb{R}^3$.

3.4. *Loss functions*

For a given set of input and ground truth flow fields, the generator model predicts flow fields that minimize a total loss function that is a combination of five specific loss functions as follows:

$$\begin{aligned}\mathcal{L}_{generator} = \frac{1}{N}\sum_{k=0}^{N-1} \{\lambda_{l2}\mathcal{L}_2^k + \lambda_{gdl}\mathcal{L}_{gdl}^k + \\ \lambda_c\mathcal{L}_c^k + \lambda_{mom}\mathcal{L}_{mom}^k + \lambda_{adv}\mathcal{L}_{adv}^{G,k}\},\end{aligned} \tag{3.7}$$

where $N(=4)$ is the number of scales of the multi-scale CNN and

$$\lambda_{l2} + \lambda_{gdl} + \lambda_c + \lambda_{mom} + \lambda_{adv} = 1. \tag{3.8}$$

Contributions of each loss function can be controlled by tuning coefficients of λ_{l2}, λ_{gdl}, λ_c, λ_{mom}, and λ_{adv}.

$\mathcal{L}_2^k$ minimizes the difference between predicted flow fields and ground truth flow fields as follows:

$$\mathcal{L}_2^k = ||G_k(\mathcal{I}) - \mathcal{G}_k(\mathcal{I})||_2^2. \tag{3.9}$$

$\mathcal{L}_{gdl}^k$ is the GDL function (Mathieu *et al.* 2015), which is applied to sharpen flow fields by directly penalizing gradient differences between predicted flow fields and ground truth

flow fields as follows:

$$\mathcal{L}_{gdl}^{k} = \sum_{i=0}^{n_x-2}\sum_{j=0}^{n_y-2} \Big\{ \Big| |\mathcal{G}_k(\mathcal{I})_{(i+1,j+1)} - \mathcal{G}_k(\mathcal{I})_{(i,j+1)}| - |G_k(\mathcal{I})_{(i+1,j+1)} - G_k(\mathcal{I})_{(i,j+1)}| \Big| + \Big| |\mathcal{G}_k(\mathcal{I})_{(i+1,j+1)} - \mathcal{G}_k(\mathcal{I})_{(i+1,j)}| - |G_k(\mathcal{I})_{(i+1,j+1)} - G_k(\mathcal{I})_{(i+1,j)}| \Big| \Big\}, \tag{3.10}$$

where the subscript (i, j) indicates the grid index of a flow field, and n_x and n_y indicate the number of grid points in x and y directions, respectively. Loss functions $\mathcal{L}_2^k$ and $\mathcal{L}_{gdl}^k$ provide prior information to networks that predicted flow fields should resemble ground truth flow fields. These loss functions support networks to learn fluid dynamics that corresponds to the flow field resemblance, by extracting features in a supervised manner.

Let $u_k/U_\infty, v_k/U_\infty, w_k/U_\infty$, and $p_k/\rho U_\infty^2$ be non-dimensionalized flow variables of ground truth flow fields ($\mathcal{G}_k(\mathcal{I})$) and $\widetilde{u}_k/U_\infty, \widetilde{v}_k/U_\infty, \widetilde{w}_k/U_\infty$, and $\widetilde{p}_k/\rho U_\infty^2$ be non-dimensionalized flow variables of predicted flow fields ($G_k(\mathcal{I})$). $\mathcal{L}_c$ enables networks to learn mass conservation by minimizing the total absolute sum of mass flux difference in each cell in an $x-y$ plane by discretizing and non-dimensionalizing Eq. (3.5) as follows:

$$\begin{aligned}\mathcal{L}_c^k = &\sum_{i=0}^{n_x-3}\sum_{j=0}^{n_y-1} \left| \left(\frac{u_{k,(i+2,j)} + u_{k,(i+1,j)}}{2U_\infty} - \frac{u_{k,(i+1,j)} + u_{k,(i,j)}}{2U_\infty} \right) \right. \\ &\left. - \left(\frac{\widetilde{u}_{k,(i+2,j)} + \widetilde{u}_{k,(i+i,j)}}{2U_\infty} - \frac{\widetilde{u}_{k,(i+1,j)} + \widetilde{u}_{k,(i,j)}}{2U_\infty} \right) \right| \\ &+ \sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3} \left| \left(\frac{v_{k,(i,j+2)} + v_{k,(i,j+1)}}{2U_\infty} - \frac{v_{k,(i,j+1)} + v_{k,(i,j)}}{2U_\infty} \right) \right. \\ &\left. - \left(\frac{\widetilde{v}_{k,(i,j+2)} + \widetilde{v}_{k,(i,j+1)}}{2U_\infty} - \frac{\widetilde{v}_{k,(i,j+1)} + \widetilde{v}_{k,(i,j)}}{2U_\infty} \right) \right|. \end{aligned} \tag{3.11}$$

$\mathcal{L}_{mom}$ enables networks to learn momentum conservation by minimizing the total absolute sum of the difference of momentum transportation due to convection, pressure, and shear stress in each cell in an $x-y$ plane by discretizing and non-dimensionalizing Eq. (3.6) as follows:

$$\begin{aligned}\mathcal{L}_{mom}^k = &\sum_{i=0}^{n_x-3}\sum_{j=0}^{n_y-1} \left| \left(\frac{u_{k,(i+2,j)} + u_{k,(i+1,j)}}{2U_\infty} \right)^2 - \left(\frac{u_{k,(i+1,j)} + u_{k,(i,j)}}{2U_\infty} \right)^2 \right. \\ &\left. - \left(\frac{\widetilde{u}_{k,(i+2,j)} + \widetilde{u}_{k,(i+1,j)}}{2U_\infty} \right)^2 + \left(\frac{\widetilde{u}_{k,(i+1,j)} + \widetilde{u}_{k,(i,j)}}{2U_\infty} \right)^2 \right| + \\ &\sum_{i=0}^{n_x-3}\sum_{j=0}^{n_y-1} \left| \left(\frac{u_{k,(i+2,j)} + u_{k,(i+1,j)}}{2U_\infty} \right) \left(\frac{v_{k,(i+2,j)} + v_{k,(i+1,j)}}{2U_\infty} \right) \right. \\ &\left. - \left(\frac{u_{k,(i+1,j)} + u_{k,(i,j)}}{2U_\infty} \right) \left(\frac{v_{k,(i+1,j)} + v_{k,(i,j)}}{2U_\infty} \right) \right. \\ &\left. - \left(\frac{\widetilde{u}_{k,(i+2,j)} + \widetilde{u}_{k,(i+1,j)}}{2U_\infty} \right) \left(\frac{\widetilde{v}_{k,(i+2,j)} + \widetilde{v}_{k,(i+1,j)}}{2U_\infty} \right) \right| \end{aligned}$$

$$
\begin{aligned}
&+\left(\frac{\widetilde{u}_{k,(i+1,j)}+\widetilde{u}_{k,(i,j)}}{2U_\infty}\right)\left(\frac{\widetilde{v}_{k,(i+1,j)}+\widetilde{v}_{k,(i,j)}}{2U_\infty}\right)\Bigg| \\
&+\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3}\Bigg|\left(\frac{v_{k,(i,j+2)}+v_{k,(i,j+1)}}{2U_\infty}\right)^2-\left(\frac{v_{k,(i,j+1)}+v_{k,(i,j)}}{2U_\infty}\right)^2 \\
&-\left(\frac{\widetilde{v}_{k,(i,j+2)}+\widetilde{v}_{k,(i,j+1)}}{2U_\infty}\right)^2+\left(\frac{\widetilde{v}_{k,(i,j+1)}+\widetilde{v}_{k,(i,j)}}{2U_\infty}\right)^2\Bigg| \\
&+\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3}\Bigg|\left(\frac{v_{k,(i,j+2)}+v_{k,(i,j+1)}}{2U_\infty}\right)\left(\frac{u_{k,(i,j+2)}+u_{k,(i,j+1)}}{2U_\infty}\right) \\
&-\left(\frac{v_{k,(i,j+1)}+v_{k,(i,j)}}{2U_\infty}\right)\left(\frac{u_{k,(i,j+1)}+u_{k,(i,j)}}{2U_\infty}\right) \\
&-\left(\frac{\widetilde{v}_{k,(i,j+2)}+\widetilde{v}_{k,(i,j+1)}}{2U_\infty}\right)\left(\frac{\widetilde{u}_{k,(i,j+2)}+\widetilde{u}_{k,(i,j+1)}}{2U_\infty}\right) \\
&-\left(\frac{\widetilde{v}_{k,(i,j+1)}+\widetilde{v}_{k,(i,j)}}{2U_\infty}\right)\left(\frac{\widetilde{u}_{k,(i,j+1)}+\widetilde{u}_{k,(i,j)}}{2U_\infty}\right)\Bigg| \\
&+\sum_{i=0}^{n_x-3}\sum_{j=0}^{n_y-1}\Bigg|\left(\frac{p_{k,(i+2,j)}+p_{k,(i+1,j)}}{2\rho U_\infty^2}\right)-\left(\frac{p_{k,(i+1,j)}+p_{k,(i,j)}}{2\rho U_\infty^2}\right) \\
&-\left(\frac{\widetilde{p}_{k,(i+2,j)}+\widetilde{p}_{k,(i+1,j)}}{2\rho U_\infty^2}\right)+\left(\frac{\widetilde{p}_{k,(i+1,j)}+\widetilde{p}_{k,(i,j)}}{2\rho U_\infty^2}\right)\Bigg| \\
&+\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3}\Bigg|\left(\frac{p_{k,(i,j+2)}+p_{k,(i,j+1)}}{2\rho U_\infty^2}\right)-\left(\frac{p_{k,(i,j+1)}+p_{k,(i,j)}}{2\rho U_\infty^2}\right) \\
&-\left(\frac{\widetilde{p}_{k,(i,j+2)}+\widetilde{p}_{k,(i,j+1)}}{2\rho U_\infty^2}\right)+\left(\frac{\widetilde{p}_{k,(i,j+1)}+\widetilde{p}_{k,(i,j)}}{2\rho U_\infty^2}\right)\Bigg|+ \\
&+\frac{1}{Re_D}\sum_{i=0}^{n_x-3}\sum_{j=0}^{n_y-1}\Bigg|\left(\frac{v_{k,(i+2,j)}-v_{k,(i+1,j)}}{U_\infty\Delta_x}+\frac{v_{k,(i,j)}-v_{k,(i+1,j)}}{U_\infty\Delta_x}\right) \\
&-\left(\frac{\widetilde{v}_{k,(i+2,j)}-\widetilde{v}_{k,(i+1,j)}}{U_\infty\Delta_x}+\frac{\widetilde{v}_{k,(i,j)}-\widetilde{v}_{k,(i+1,j)}}{U_\infty\Delta_x}\right)\Bigg| \\
&+\frac{1}{Re_D}\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3}\Bigg|\left(\frac{v_{k,(i,j+2)}-v_{k,(i,j+1)}}{U_\infty\Delta_y}+\frac{v_{k,(i,j)}-v_{k,(i,j+1)}}{U_\infty\Delta_y}\right) \\
&-\left(\frac{\widetilde{v}_{k,(i,j+2)}-\widetilde{v}_{k,(i,j+1)}}{U_\infty\Delta_y}+\frac{\widetilde{v}_{k,(i,j)}-\widetilde{v}_{k,(i,j+1)}}{U_\infty\Delta_y}\right)\Bigg| \\
&+\frac{1}{Re_D}\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3}\Bigg|\left(\frac{u_{k,(i,j+2)}-u_{k,(i,j+1)}}{U_\infty\Delta_y}+\frac{u_{k,(i,j)}-u_{k,(i,j+1)}}{U_\infty\Delta_y}\right) \\
&-\left(\frac{\widetilde{u}_{k,(i,j+2)}-\widetilde{u}_{k,(i,j+1)}}{U_\infty\Delta_y}+\frac{\widetilde{u}_{k,(i,j)}-\widetilde{u}_{k,(i,j+1)}}{U_\infty\Delta_y}\right)\Bigg| \\
&+\frac{1}{Re_D}\sum_{i=0}^{n_x-3}\sum_{j=0}^{n_y-1}\Bigg|\left(\frac{u_{k,(i+2,j)}-u_{k,(i+1,j)}}{U_\infty\Delta_x}+\frac{u_{k,(i,j)}-u_{k,(i+1,j)}}{U_\infty\Delta_x}\right)
\end{aligned}
$$

$$-\left(\frac{\widetilde{u}_{k,(i+2,j)}-\widetilde{u}_{k,(i+1,j)}}{U_\infty \Delta_x}+\frac{\widetilde{u}_{k,(i,j)}-\widetilde{u}_{k,(i+1,j)}}{U_\infty \Delta_x}\right)\Bigg|, \quad (3.12)$$

where Δ_x and Δ_y are grid spacings in x and y directions, respectively. Loss functions $\mathcal{L}_c$ and $\mathcal{L}_{mom}$, which are defined as physical loss functions, provide explicit prior information of physical conservation laws to networks, and support networks to extract features including physical conservation laws in a supervised manner.

$\mathcal{L}^G_{adv}$ is a loss function with purpose to delude the discriminator model to classify generated flow fields to class 1 as

$$\mathcal{L}^G_{adv} = L_{bce}(D_k(G_k(\mathcal{I})), 1), \quad (3.13)$$

where L_{bce} is the binary cross entropy loss function defined as follows:

$$L_{bce}(a, b) = -b\log(a) - (1-b)\log(1-a), \quad (3.14)$$

for scalar values a and b between 0 and 1. The loss function $\mathcal{L}^G_{adv}$ provides knowledge in a concealed manner that features of the predicted and ground truth flow fields should be indistinguishable. This loss function supports networks to extract features of underlying fluid dynamics in an unsupervised manner.

The loss function of the discriminator model is defined as follows:

$$\mathcal{L}_{discriminator} = \frac{1}{N}\sum_{k=0}^{N-1}\left[L_{bce}(D_k(\mathcal{G}_k(\mathcal{I})), 1) + L_{bce}(D_k(G_k(\mathcal{I})), 0)\right]. \quad (3.15)$$

$\mathcal{L}_{discriminator}$ is minimized so that the discriminator model appropriately classifies ground truth flow fields into class 1 and predicted flow fields into class 0. The discriminator model learns flow fields in the low-dimensional feature space.

4. Prediction and evaluation

4.1. *Error functions for evaluation*

Let $u/U_\infty, v/U_\infty, w/U_\infty$, and $p/\rho U_\infty^2$ be non-dimensionalized flow variables of the ground truth flow field $(\mathcal{G}(\mathcal{I}))$ and $\widetilde{u}/U_\infty, \widetilde{v}/U_\infty, \widetilde{w}/U_\infty$, and $\widetilde{p}/\rho U_\infty^2$ be non-dimensionalized flow variables of the generated flow field $(G(\mathcal{I}))$. An L_2 error of the prediction is evaluated based on an average difference in flow variables as follows:

$$L_2 = \left(\frac{1}{4n_x n_y}\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-1}\{(\frac{u_{(i,j)}-\widetilde{u}_{(i,j)}}{U_\infty})^2 + (\frac{v_{(i,j)}-\widetilde{v}_{(i,j)}}{U_\infty})^2 + (\frac{w_{(i,j)}-\widetilde{w}_{(i,j)}}{U_\infty})^2 + (\frac{p_{(i,j)}-\widetilde{p}_{(i,j)}}{\rho U_\infty^2})^2\}\right)^{1/2}. \quad (4.1)$$

L_∞ error of the prediction is evaluated based on an average of the maximum difference in flow variables as follows:

$$L_\infty = \frac{1}{4}\left(\max_{i,j}(\frac{u_{(i,j)}-\widetilde{u}_{(i,j)}}{U_\infty}) + \max_{i,j}(\frac{v_{(i,j)}-\widetilde{v}_{(i,j)}}{U_\infty}) + \max_{i,j}(\frac{w_{(i,j)}-\widetilde{w}_{(i,j)}}{U_\infty}) + \max_{i,j}(\frac{p_{(i,j)}-\widetilde{p}_{(i,j)}}{\rho U_\infty^2})\right). \quad (4.2)$$

The error in mass conservation is evaluated based on an average difference of mass

fluxes at streamwise and cross-flow directions in each grid cell as follows:

$$L_c = \frac{1}{2n_y(n_x-2)} \sum_{i=0}^{n_x-3} \sum_{j=0}^{n_y-1} \left| \left(\frac{u_{(i+2,j)} + u_{(i+1,j)}}{2U_\infty} - \frac{u_{(i+1,j)} + u_{(i,j)}}{2U_\infty} \right) - \left(\frac{\widetilde{u}_{(i+2,j)} + \widetilde{u}_{(i+i,j)}}{2U_\infty} - \frac{\widetilde{u}_{(i+1,j)} + \widetilde{u}_{(i,j)}}{2U_\infty} \right) \right| + \frac{1}{2n_x(n_y-2)} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-3} \left| \left(\frac{v_{(i,j+2)} + v_{(i,j+1)}}{2U_\infty} - \frac{v_{(i,j+1)} + v_{(i,j)}}{2U_\infty} \right) - \left(\frac{\widetilde{v}_{(i,j+2)} + \widetilde{v}_{(i,j+1)}}{2U_\infty} - \frac{\widetilde{v}_{(i,j+1)} + \widetilde{v}_{(i,j)}}{2U_\infty} \right) \right|. \tag{4.3}$$

The error in momentum conservation is evaluated based on an average difference of momentum fluxes due to convection, pressure, and shear stress in each grid cell as follows:

$$\begin{aligned} L_{mom} = {} & \frac{1}{3n_y(n_x-2)} \sum_{i=0}^{n_x-3} \sum_{j=0}^{n_y-1} \left| \left(\frac{u_{(i+2,j)} + u_{(i+1,j)}}{2U_\infty} \right)^2 - \left(\frac{u_{(i+1,j)} + u_{(i,j)}}{2U_\infty} \right)^2 \right. \\ & \left. - \left(\frac{\widetilde{u}_{(i+2,j)} + \widetilde{u}_{(i+1,j)}}{2U_\infty} \right)^2 + \left(\frac{\widetilde{u}_{(i+1,j)} + \widetilde{u}_{(i,j)}}{2U_\infty} \right)^2 \right| \\ & + \frac{1}{3n_y(n_x-2)} \sum_{i=0}^{n_x-3} \sum_{j=0}^{n_y-1} \left| \left(\frac{u_{(i+2,j)} + u_{(i+1,j)}}{2U_\infty} \right) \left(\frac{v_{(i+2,j)} + v_{(i+1,j)}}{2U_\infty} \right) \right. \\ & - \left(\frac{u_{(i+1,j)} + u_{(i,j)}}{2U_\infty} \right) \left(\frac{v_{(i+1,j)} + v_{(i,j)}}{2U_\infty} \right) \\ & - \left(\frac{\widetilde{u}_{(i+2,j)} + \widetilde{u}_{(i+1,j)}}{2U_\infty} \right) \left(\frac{\widetilde{v}_{(i+2,j)} + \widetilde{v}_{(i+1,j)}}{2U_\infty} \right) \\ & \left. + \left(\frac{\widetilde{u}_{(i+1,j)} + \widetilde{u}_{(i,j)}}{2U_\infty} \right) \left(\frac{\widetilde{v}_{(i+1,j)} + \widetilde{v}_{(i,j)}}{2U_\infty} \right) \right| \\ & + \frac{1}{3n_x(n_y-2)} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-3} \left| \left(\frac{v_{(i,j+2)} + v_{(i,j+1)}}{2U_\infty} \right)^2 - \left(\frac{v_{(i,j+1)} + v_{(i,j)}}{2U_\infty} \right)^2 \right. \\ & \left. - \left(\frac{\widetilde{v}_{(i,j+2)} + \widetilde{v}_{(i,j+1)}}{2U_\infty} \right)^2 + \left(\frac{\widetilde{v}_{(i,j+1)} + \widetilde{v}_{(i,j)}}{2U_\infty} \right)^2 \right| \\ & + \frac{1}{3n_x(n_y-2)} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-3} \left| \left(\frac{v_{(i,j+2)} + v_{(i,j+1)}}{2U_\infty} \right) \left(\frac{u_{(i,j+2)} + u_{(i,j+1)}}{2U_\infty} \right) \right. \\ & - \left(\frac{v_{(i,j+1)} + v_{(i,j)}}{2U_\infty} \right) \left(\frac{u_{(i,j+1)} + u_{(i,j)}}{2U_\infty} \right) \\ & - \left(\frac{\widetilde{v}_{(i,j+2)} + \widetilde{v}_{(i,j+1)}}{2U_\infty} \right) \left(\frac{\widetilde{u}_{(i,j+2)} + \widetilde{u}_{(i,j+1)}}{2U_\infty} \right) \\ & \left. - \left(\frac{\widetilde{v}_{(i,j+1)} + \widetilde{v}_{(i,j)}}{2U_\infty} \right) \left(\frac{\widetilde{u}_{(i,j+1)} + \widetilde{u}_{(i,j)}}{2U_\infty} \right) \right| \\ & + \frac{1}{3n_y(n_x-2)} \sum_{i=0}^{n_x-3} \sum_{j=0}^{n_y-1} \left| \left(\frac{p_{(i+2,j)} + p_{(i+1,j)}}{2\rho U_\infty^2} \right) - \left(\frac{p_{(i+1,j)} + p_{(i,j)}}{2\rho U_\infty^2} \right) \right. \end{aligned}$$

$$-\left(\frac{\widetilde{p}_{(i+2,j)}+\widetilde{p}_{(i+1,j)}}{2\rho U_\infty^2}\right)+\left(\frac{\widetilde{p}_{(i+1,j)}+\widetilde{p}_{(i,j)}}{2\rho U_\infty^2}\right)\Bigg|$$

$$+\frac{1}{3n_x(n_y-2)}\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3}\Bigg|\left(\frac{p_{(i,j+2)}+p_{(i,j+1)}}{2\rho U_\infty^2}\right)-\left(\frac{p_{(i,j+1)}+p_{(i,j)}}{2\rho U_\infty^2}\right)$$

$$-\left(\frac{\widetilde{p}_{(i,j+2)}+\widetilde{p}_{(i,j+1)}}{2\rho U_\infty^2}\right)+\left(\frac{\widetilde{p}_{(i,j+1)}+\widetilde{p}_{(i,j)}}{2\rho U_\infty^2}\right)\Bigg|+$$

$$+\frac{1}{3Re_D n_y(n_x-2)}\sum_{i=0}^{n_x-3}\sum_{j=0}^{n_y-1}\Bigg|\left(\frac{v_{(i+2,j)}-v_{(i+1,j)}}{U_\infty\Delta_x}+\frac{v_{(i,j)}-v_{(i+1,j)}}{U_\infty\Delta_x}\right)$$

$$-\left(\frac{\widetilde{v}_{(i+2,j)}-\widetilde{v}_{(i+1,j)}}{U_\infty\Delta_x}+\frac{\widetilde{v}_{(i,j)}-\widetilde{v}_{(i+1,j)}}{U_\infty\Delta_x}\right)\Bigg|$$

$$+\frac{1}{3Re_D n_x(n_y-2)}\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3}\Bigg|\left(\frac{v_{(i,j+2)}-v_{(i,j+1)}}{U_\infty\Delta_y}+\frac{v_{(i,j)}-v_{(i,j+1)}}{U_\infty\Delta_y}\right)$$

$$-\left(\frac{\widetilde{v}_{(i,j+2)}-\widetilde{v}_{(i,j+1)}}{U_\infty\Delta_y}+\frac{\widetilde{v}_{(i,j)}-\widetilde{v}_{(i,j+1)}}{U_\infty\Delta_y}\right)\Bigg|$$

$$+\frac{1}{3Re_D n_x(n_y-2)}\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-3}\Bigg|\left(\frac{u_{(i,j+2)}-u_{(i,j+1)}}{U_\infty\Delta_y}+\frac{u_{(i,j)}-u_{(i,j+1)}}{U_\infty\Delta_y}\right)$$

$$-\left(\frac{\widetilde{u}_{(i,j+2)}-\widetilde{u}_{(i,j+1)}}{U_\infty\Delta_y}+\frac{\widetilde{u}_{(i,j)}-\widetilde{u}_{(i,j+1)}}{U_\infty\Delta_y}\right)\Bigg|$$

$$+\frac{1}{3Re_D n_y(n_x-2)}\sum_{i=0}^{n_x-3}\sum_{j=0}^{n_y-1}\Bigg|\left(\frac{u_{(i+2,j)}-u_{(i+1,j)}}{U_\infty\Delta_x}+\frac{u_{(i,j)}-u_{(i+1,j)}}{U_\infty\Delta_x}\right)$$

$$-\left(\frac{\widetilde{u}_{(i+2,j)}-\widetilde{u}_{(i+1,j)}}{U_\infty\Delta_x}+\frac{\widetilde{u}_{(i,j)}-\widetilde{u}_{(i+1,j)}}{U_\infty\Delta_x}\right)\Bigg|. \quad (4.4)$$

4.2. *Comparison of loss functions*

Four deep learning networks with different combinations of coefficients for loss functions (λ) are trained (see table 3). Case 1 utilizes CNNs without physical loss functions; Case 2 utilizes CNNs with physical loss functions; Case 3 utilizes a GAN without physical loss functions; Case 4 utilizes a GAN with physical loss functions. Each deep learning case is trained with flow fields at $Re_D = 100, 200, 300$, and 400 using a generator model with the configuration of GM_{20} and number set N_{128} (see table 1), while cases 3 and 4 are also trained with a discriminator model (see table 2).

Contour plots of flow fields predicted by the four different deep learning networks for flow at $Re_D = 500$ and 3000 are shown in figures 7-14. Flow fields after a time-step interval larger than δt are predicted recursively by utilizing flow fields predicted prior time-steps as parts of the input. Predicted flow fields after δt from all deep learning cases are found to agree well with ground truth flow fields, even the trained networks have not seen such small-scale flow structures at higher Reynolds numbers. However, differences between the predicted and ground truth flow fields increase as the recursive step increases because errors from the previous predictions are accumulated to the next time-step prediction. Particularly, dissipation of small-scale flow-structures are observed; however, movements of large-scale flow structures from the vortex shedding are favorably predicted.

L_2 and L_∞ errors evaluated from each deep learning case are compared in figure 15.

Case	λ_{l2}	λ_{gdl}	λ_c	λ_{mom}	λ_{adv}
1	0.5	0.5	0.0	0.0	0.0
2	0.25	0.25	0.25	0.25	0.0
3	0.4625	0.4625	0.0	0.0	0.075
4	0.23125	0.23125	0.23125	0.23125	0.075

Table 3: Coefficients of loss functions of each deep learning case.

Case	λ_{adv}	λ_{l2}	λ_{gdl}
A	0.025	0.4875	0.4875
B	0.05	0.475	0.475
C	0.075	0.4625	0.4625
D	0.1	0.45	0.45

Table 4: Coefficients of loss functions for calculating adversarial training effects. λ_c, λ_{mom}, and λ_{ke} are zero.

For all four cases (cases 1-4), as shown in figures 15(a), (c), and (e), similar levels of L_2 errors are obtained for both $Re_D = 500$ and 3000, while noticeable differences in L_∞ errors among four cases are observed (figures 15(b), (d), and (f)). Case 3 is found to produce lowest L_∞ errors during recursive prediction of flow fields at both $Re_D = 500$ and 3000 (see figures 15(d) and (f)). Note that the L_∞ error is not explicitly minimized by the generator loss function $\mathcal{L}_{generator}$. This result implies that a GAN employing adversarial training is best capable of extracting features in cylinder wake flow. On the other hand, case 2 shows the largest L_∞ errors and standard deviations on the predicted flow fields at both $Re_D = 500$ and 3000. The CNN with weights on physical loss functions (λ_c and λ_{mom}) while with reduced weights on similarity loss functions (λ_{l2} and λ_{gdl}) is found to be less accurate in predicting future flow fields. Similarly, case 4 exhibits relatively large L_∞ errors compared to cases 1 and 3 which have larger weights on loss functions for resemblance (see table 3).

L_c and L_{mom} errors evaluated from each deep learning case are compared in figure 16. As shown in figures 16(a) and (b), case 4 shows the minimum errors in the flow field prediction after δt. During recursive predictions, L_c and L_{mom} errors are found to be noticeably improved in cases 2 (see figures 16(c)-(f)), and this result indicates that the incorporation of physical loss functions improves the conservation of mass and momentum.

4.3. *Effects of the adversarial loss function*

Effects of the adversarial loss function on the prediction are analyzed in detail by tuning the coefficient (λ_{adv}) in four different cases (cases A-D, see table 4). A generator

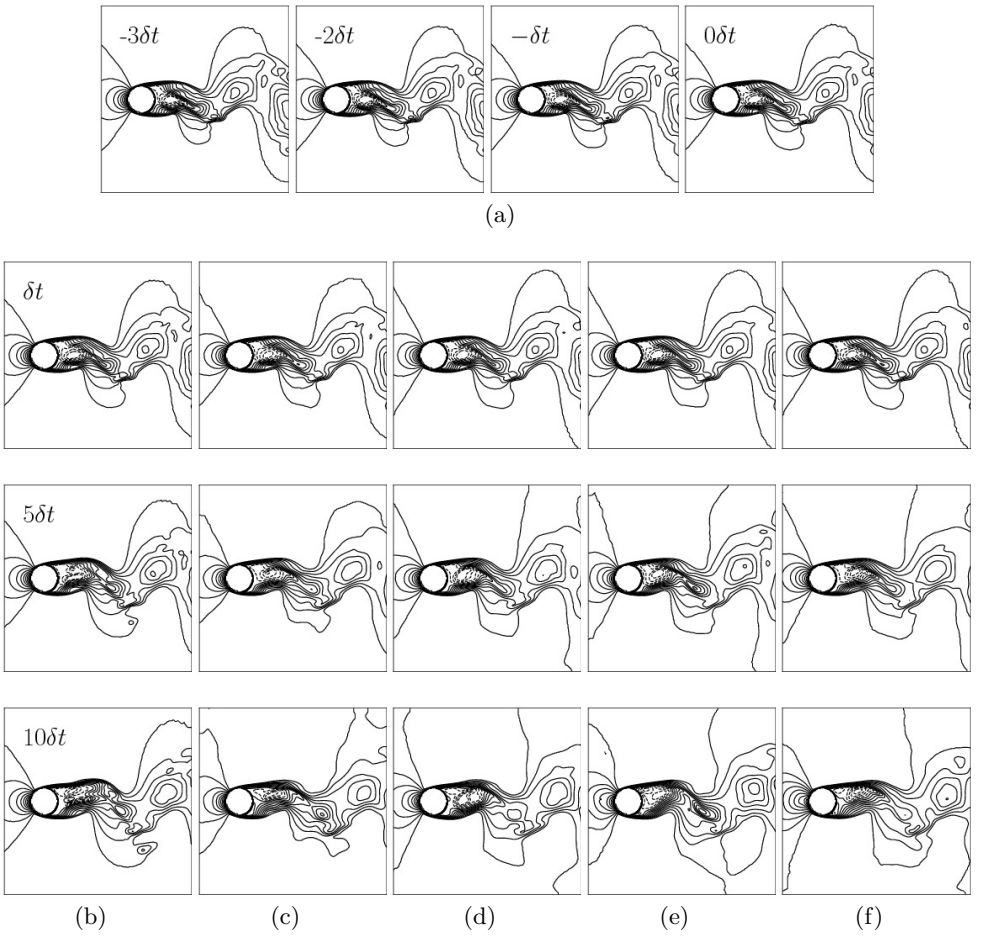


Figure 7: Comparison of the streamwise velocity (u/U_∞) at $Re_D = 500$ after δt, $5\delta t$, and $10\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $5\delta t$ and $10\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-column: ground truth; (c)-column: predicted in Case 1; (d)-column: predicted in Case 2; (e)-column: predicted in Case 3; (f)-column: predicted in Case 4. 15 contour levels from -0.5 to 1.0 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

model with the configuration of GM_{20} and number set N_{128} and a discriminator model are trained with flow fields at $Re_D = 100, 200, 300$, and 400 for each case. Case C shows the minimum L_2 errors of flow field predictions at $Re_D = 3000$ (figure 17(a)). This indicates that a moderately large weight on adversarial training supports a network to extract general flow features that are also contained in high Reynolds number flow over a circular cylinder. In addition, as shown in figures 17(c) and (d), cases with high weights on adversarial training (cases C and D) show smaller errors of L_c and L_{mom} at both $Re_D = 500$ and 3000 compared to cases with low weights on adversarial training (cases A and B). Therefore, a high weight on adversarial training supports the network to extract flow features containing physical conservation laws in an unsupervised manner.

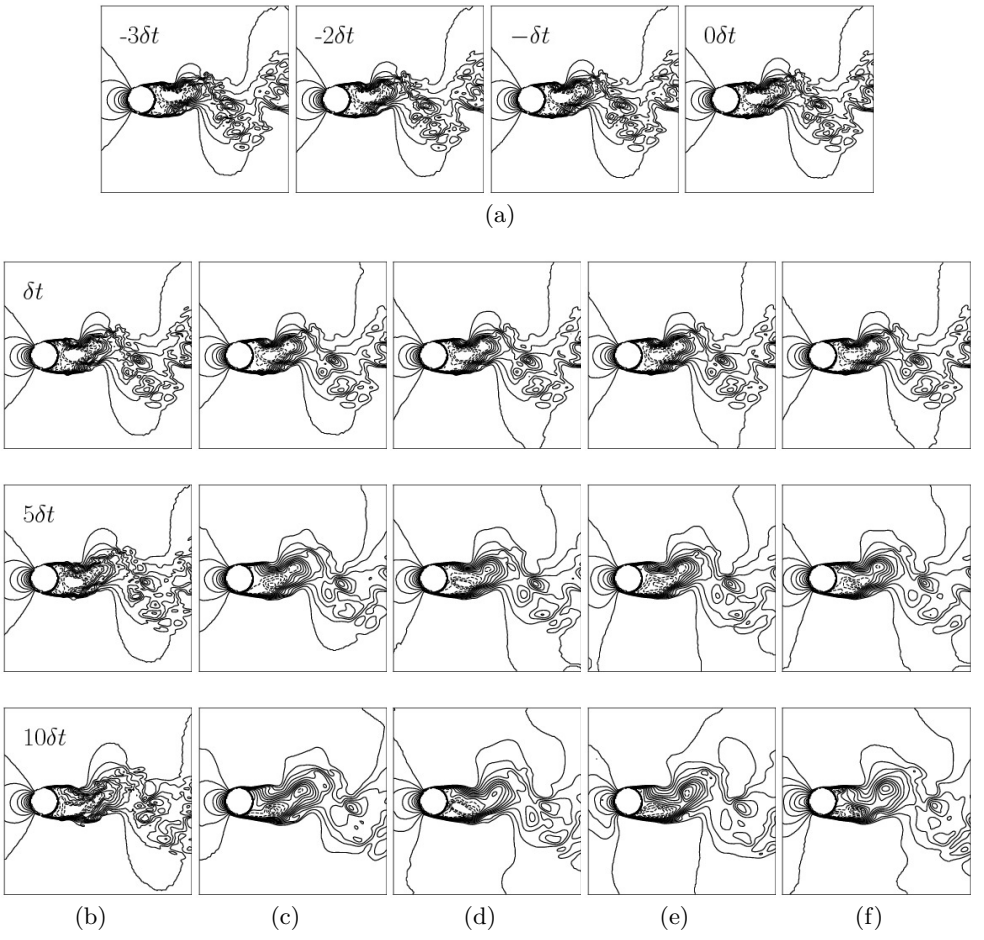


Figure 8: Comparison of the streamwise velocity (u/U_∞) at $Re_D = 3000$ after δt, $5\delta t$, and $10\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $5\delta t$ and $10\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-column: ground truth; (c)-column: predicted in Case 1; (d)-column: predicted in Case 2; (e)-column: predicted in Case 3; (f)-column: predicted in Case 4. 15 contour levels from -0.5 to 1.0 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

Unsupervised feature extraction using adversarial training provides potential to discover and learn uncovered information in data.

4.4. *Effects of the feature map size and the number of layers in the generator model*

Effects of the feature map size of the generator model in case 1 are analyzed by training the generator model with number sets N_{32}, N_{64}, and N_{128} with the configuration of GM_{20}. A larger number of feature maps allows a network to learn higher dimensional representation of data. As a consequence, as shown in figure 18, the generator model with the largest number set, N_{128}, shows significantly reduced errors compared to smaller number sets (N_{32} and N_{64}) for both Reynolds numbers of 500 and 3000.

Effects of the number of convolution layers of the generator model in case 1 are analyzed

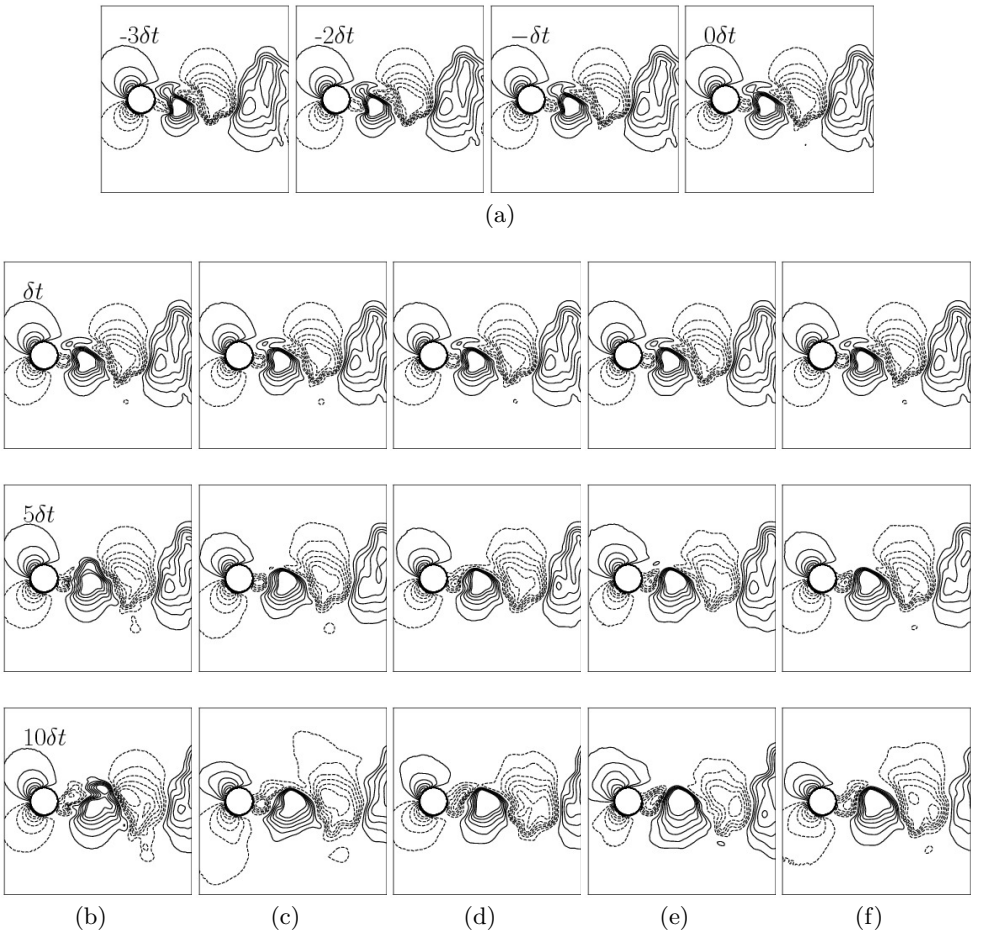


Figure 9: Comparison of the cross-stream velocity (v/U_∞) at $Re_D = 500$ after δt, $5\delta t$, and $10\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $5\delta t$ and $10\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-column: ground truth; (c)-column: predicted in Case 1; (d)-column: predicted in Case 2; (e)-column: predicted in Case 3; (f)-column: predicted in Case 4. 15 contour levels from -0.7 to 0.7 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

in detail by training three generator models with the configuration of GM_{16}, GM_{18}, and GM_{20} with the number set of N_{128}. As shown in figure 19, the configuration with the largest number of convolution layers (GM_{20}) shows the smallest errors compared to configurations with smaller numbers of convolution layers (GM_{16} and $GM18$). However, errors from the configuration of GM_{16} is smaller than errors from the configuration of GM_{18}, which contains a larger number of convolution layers. Although a large number of convolution layers supports a network to learn or extract non-linear features, no specific trends are observed with respect to the number of the convolution layers considered in the present generator models.

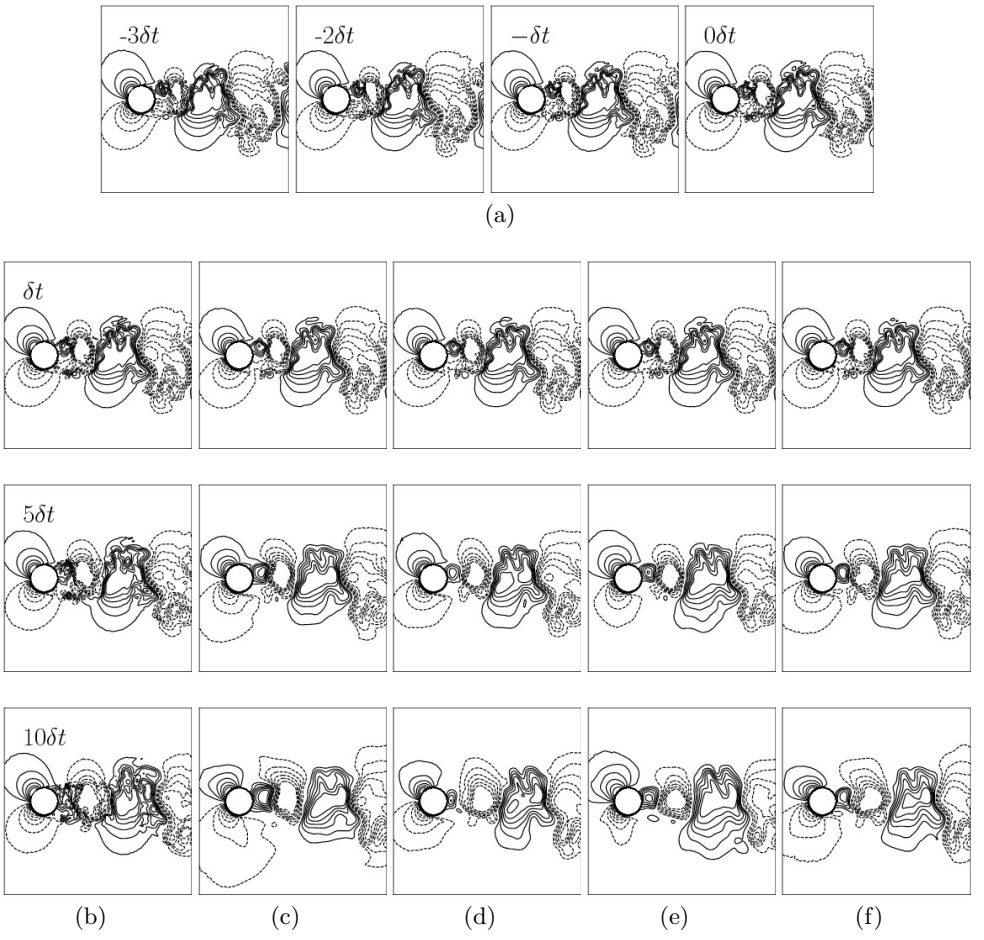


Figure 10: Comparison of the cross-stream velocity (v/U_∞) at $Re_D = 3000$ after δt, $5\delta t$, and $10\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $5\delta t$ and $10\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-column: ground truth; (c)-column: predicted in Case 1; (d)-column: predicted in Case 2; (e)-column: predicted in Case 3; (f)-column: predicted in Case 4. 15 contour levels from -0.7 to 0.7 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

4.5. *Prediction in large time-step intervals*

To investigate the potential of using a GAN in practical applications, where predicting large-scale flow motions is important, the GAN without physical loss functions (case 3) is trained with a large time-step interval size of $25\delta t = 500\Delta t U_\infty/D = 0.25$. This time-step interval is 25 times larger than the previous deep learning time-step interval and 500 times larger than the simulation time-step interval. Prediction results at $Re_D = 500$ and 3000 are shown in figures 20-27. Flow fields after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are predicted. Note that $125\delta t$ corresponds to 2500 time-step intervals of the conducted numerical simulations. As shown in figures 20 and 21, large-scale oscillations of the streamwise velocity behind the cylinder are favorably predicted. Detachment

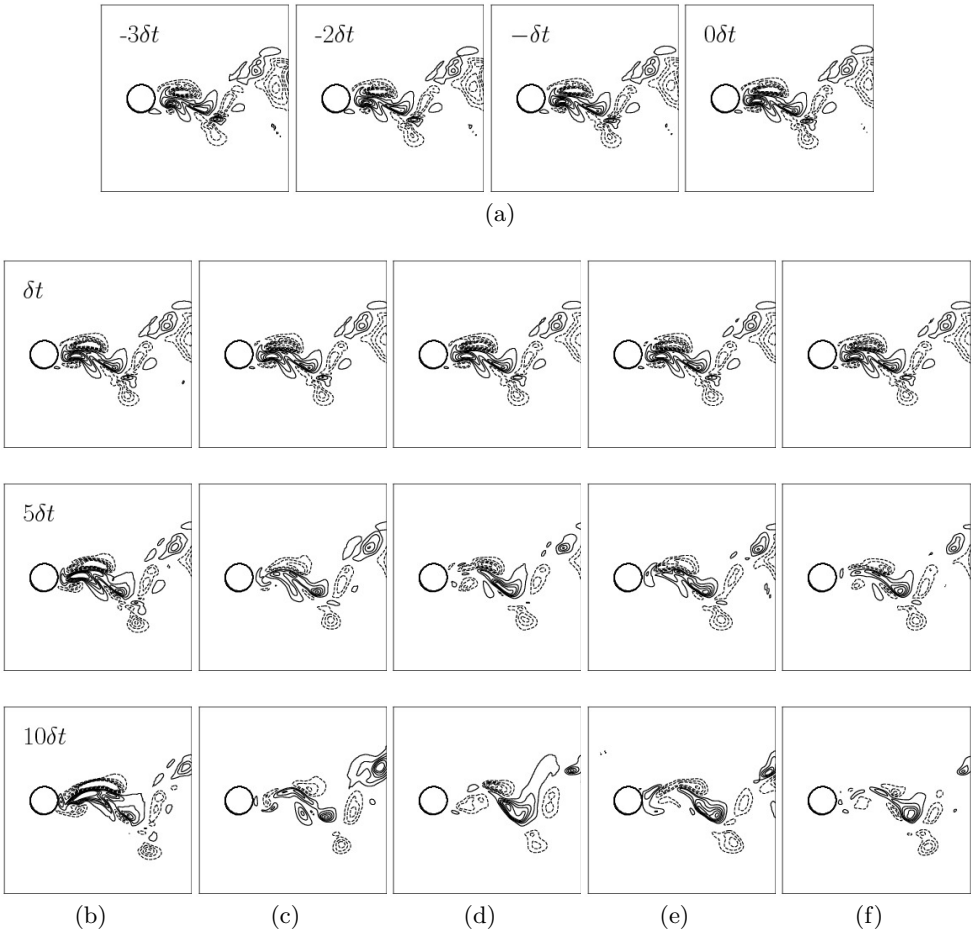


Figure 11: Comparison of the spanwise velocity (w/U_∞) at $Re_D = 500$ after δt, $5\delta t$, and $10\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $5\delta t$ and $10\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-column: ground truth; (c)-column: predicted in Case 1; (d)-column: predicted in Case 2; (e)-column: predicted in Case 3; (f)-column: predicted in Case 4. 15 contour levels from -0.5 to 0.5 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

and oscillation of separated shear layers and convection of cross-stream velocity and pressure are also well predicted (see figures 22, 23, 26, and 27). However, small-scale flow structures are found to be dissipated as especially noticeable in the prediction of the spanwise velocity (see figures 24 and 25). The generator model of the GAN is designed to predict flow fields with the highest possibility to occur in the future, based on the learned weights from a result of a minimization problem of the generator loss function. Therefore, the dissipation implies that the GAN learned convolution kernel weights from a local minimum of the loss function, which corresponds to large-scale flow motions, rather than the global minimum of the loss function, which corresponds to the exact solution of the Navier-Stokes equations including small-scale flow motions. Nevertheless,

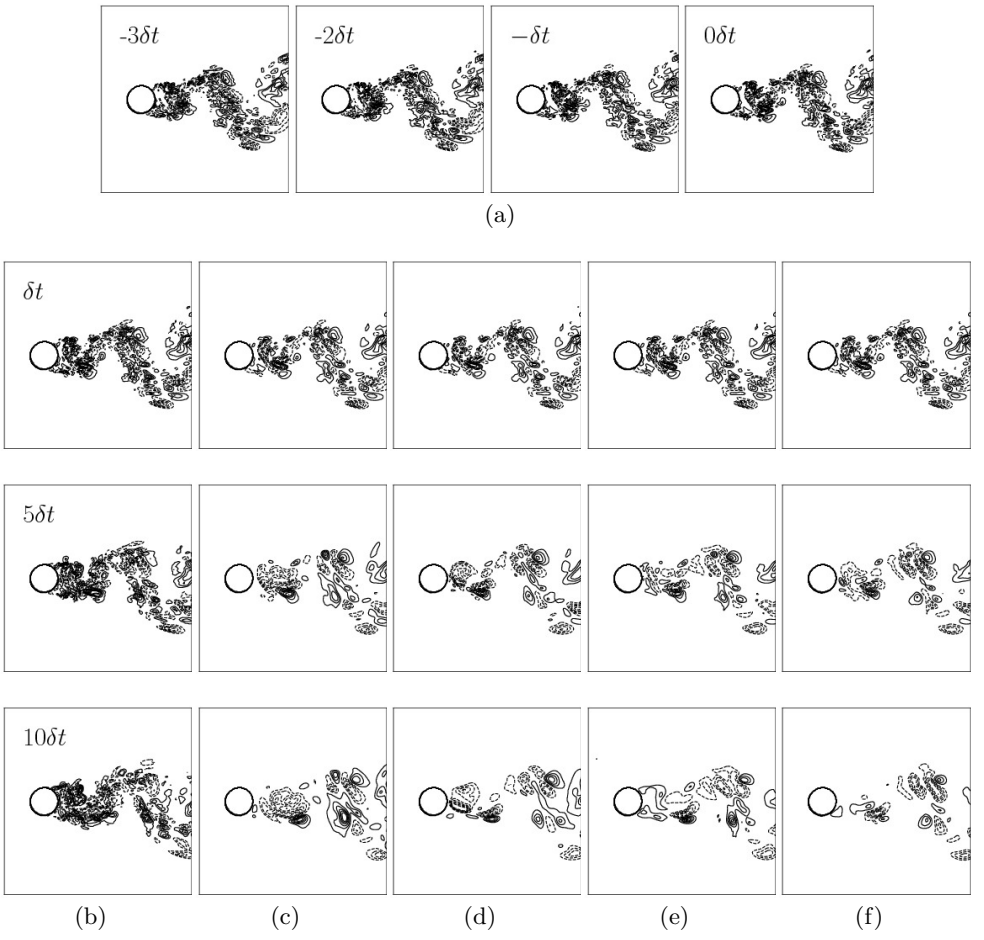


Figure 12: Comparison of the spanwise velocity (w/U_∞) at $Re_D = 3000$ after δt, $5\delta t$, and $10\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $5\delta t$ and $10\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-column: ground truth; (c)-column: predicted in Case 1; (d)-column: predicted in Case 2; (e)-column: predicted in Case 3; (f)-column: predicted in Case 4. 15 contour levels from -0.5 to 0.5 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

the GAN is found to favorably predict dominant flow motions at significantly large time intervals (several flow cycles).

L_2 errors at $Re_D = 500$ and 3000 as a function of recursive time steps are shown in figure 28. Flow predictions at $Re_D = 500$ and 3000 show similar levels of L2 errors due to the dissipation of small-scale flow motions and the well predicted large-scale flow motions on both Reynolds numbers. In contrast to errors in the recursive prediction using small time-step interval sizes (see section 4.2), L_2 errors in large time-step interval cases do not monotonically increase but show fluctuations, especially at around $100\delta t$ for both Reynolds numbers (see figure 28).

L_2 errors of each flow varibale from predictions using a small time-step interval ($\delta t = 20\Delta t U_\infty/D = 0.01$) with 25 recursive steps and a large time-step interval

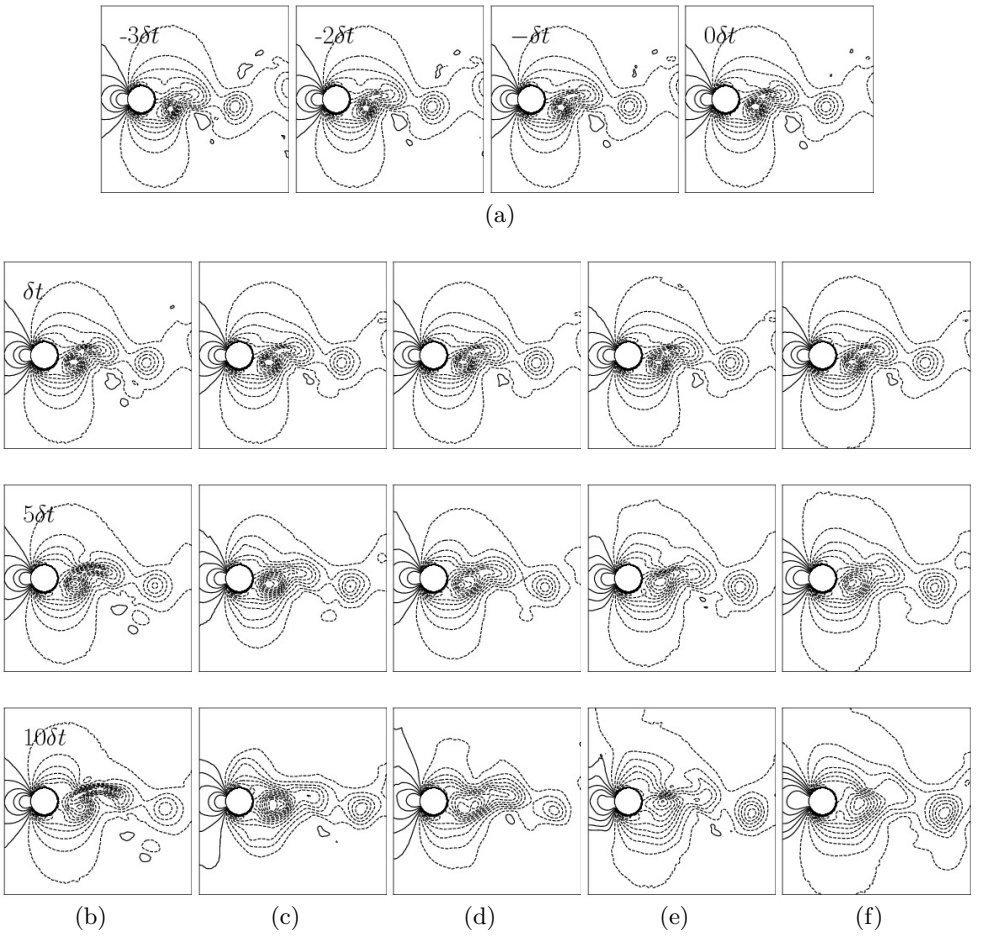


Figure 13: Comparison of the pressure ($p/\rho U_\infty^2$) at $Re_D = 500$ after δt, $5\delta t$, and $10\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $5\delta t$ and $10\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-column: ground truth; (c)-column: predicted in Case 1; (d)-column: predicted in Case 2; (e)-column: predicted in Case 3; (f)-column: predicted in Case 4. 15 contour levels from -1.0 to 0.4 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

($25\delta t = 500\Delta t U_\infty/D = 0.25$) with a single prediction step are compared in table 5. Predicting flow fields after a single prediction step with a large time-step interval size exhibits significant error reductions on flow variables u/U_∞, v/U_∞, and $p/\rho U_\infty^2$ compared to predicting flow fields using 25 recursive prediction steps with a small time-step interval size. This is because errors from previous prediction steps, which are sensitive to the dissipation of small-scale flow motions, are accumulated in the case of learning with a small time-step interval size. The errors of w/U_∞ from both time-step interval size show similar levels because the spanwise velocity is nearly dissipated in both prediction cases.

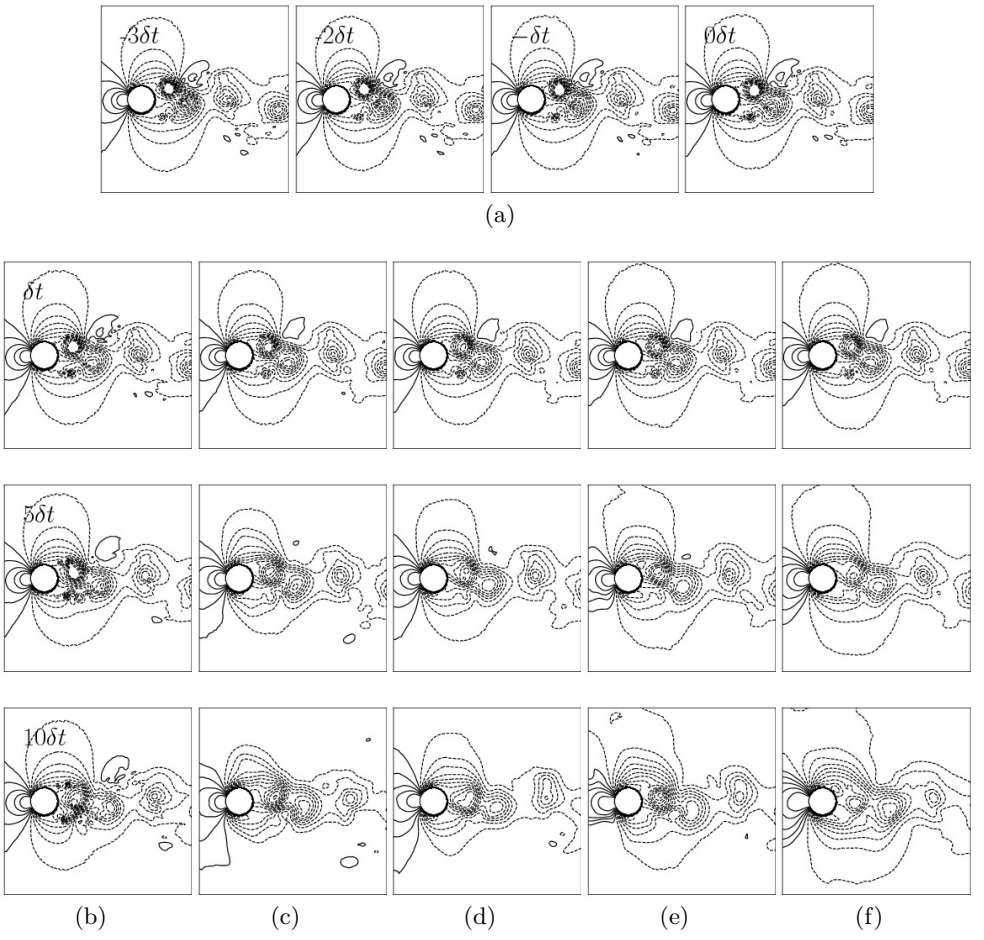


Figure 14: Comparison of the pressure ($p/\rho U_\infty^2$) at $Re_D = 3000$ after δt, $5\delta t$, and $10\delta t$, where $1\delta t = 20\Delta t U_\infty / D = 0.01$. Flow fields at $5\delta t$ and $10\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-column: ground truth; (c)-column: predicted in Case 1; (d)-column: predicted in Case 2; (e)-column: predicted in Case 3; (f)-column: predicted in Case 4. 15 contour levels from -1.0 to 0.4 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

5. Conclusions

Flow fields around a circular cylinder at $Re_D = 500$ and 3000, which are not in the training dataset, were successfully predicted using deep learning, which had been trained using flow field datasets produced by numerical simulations at $Re_D = 100, 200, 300$, and 400. Four deep learning networks (CNNs with and without physical loss functions and GANs with and without physical loss functions) with different loss functions have been trained and compared to each other.

Physical loss functions have been proposed to explicitly provide information of mass and momentum conservations to a deep learning network. For CNNs, the incorporation of physical loss functions is found to be effective in improving the conservation of mass and

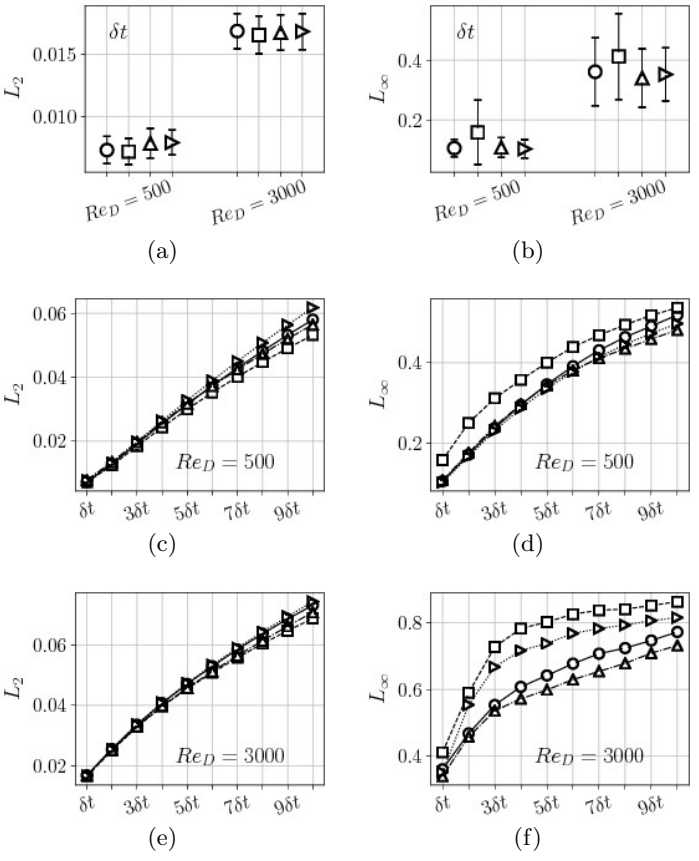


Figure 15: L_2 and L_∞ errors. (a) and (b) are the mean errors from predictions after δt. Error bars represent the standard deviation; (c) and (d) are mean errors from recursive predictions at $Re_D = 500$; (e) and (f) are mean errors from recursive predictions at $Re_D = 3000$. The time-step interval between flow fields is $\delta t = 20\Delta t U_\infty / D = 0.01$. $\circ$ and solid line: Case 1; $\square$ and dashed line: Case 2; $\triangle$ and dash-dotted line: Case 3; $\triangleright$ and dotted line: Case 4.

momentum. On the other hand, for GANs, the contribution of the adversarial training is more important than the physical loss functions to the conservation of mass and momentum.

Compared to CNNs, a GAN employing adversarial training has been found to significantly reduce L_∞ errors in recursive predictions. In addition, the GAN has been found to be capable of recursively predicting large-scale flow motions accurately with large time-step interval sizes, which are 500 times larger than the time-step size used in the numerical simulations. Such capability is expected to be useful in many practical applications, such as real-time flow control and guidance of aero- or hydro-vehicles, fast weather forecast, *etc.*, where accurate and fast prediction of energetic large-scale flow motions is important.

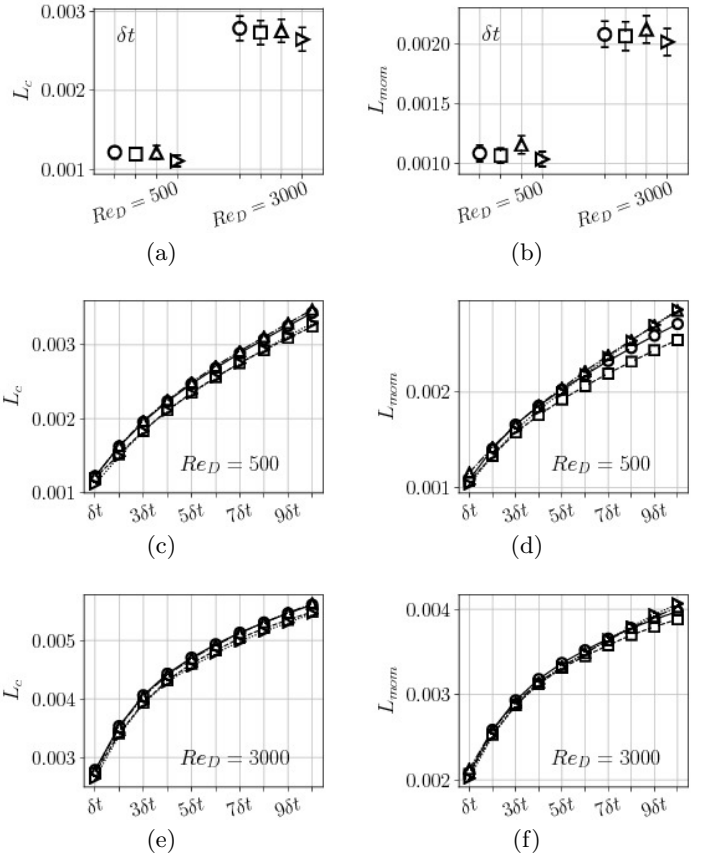


Figure 16: L_c and L_{mom} errors. (a) and (b) are the mean errors from predictions after δt. Error bars represent the standard deviation; (c) and (d) are mean errors from recursive predictions at $Re_D = 500$; (e) and (f) are mean errors from recursive predictions at $Re_D = 3000$. The time-step interval between flow fields is $\delta t = 20\Delta t U_\infty / D = 0.01$. $\circ$ and solid line: Case 1; $\square$ and dashed line: Case 2; $\triangle$ and dash-dotted line: Case 3; $\triangleright$ and dotted line: Case 4.

6. Acknowledgements

This work was supported by the Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-TB1703-01 and National Research Foundation of Korea (NRF) under Grant Number NRF-2017R1E1A 1A03070514.

REFERENCES

Babucke, Andreas, Kloker, Markus & Rist, Ulrich 2008 DNS of a plane mixing layer for the investigation of sound generation mechanisms. *Computers & Fluids* **37** (4), 360–368.

Berger, Eberhard & Wille, Rudolf 1972 Periodic flow phenomena. *Annual Review of Fluid Mechanics* **4** (1), 313–340.

Dalin, P, Pertsev, N, Frandsen, S, Hansen, O, Andersen, H, Dubietis, A & Balciunas, R 2010 A case study of the evolution of a Kelvin–Helmholtz wave and turbulence in noctilucent clouds. *Journal of Atmospheric and Solar-Terrestrial Physics* **72** (14), 1129–1138.

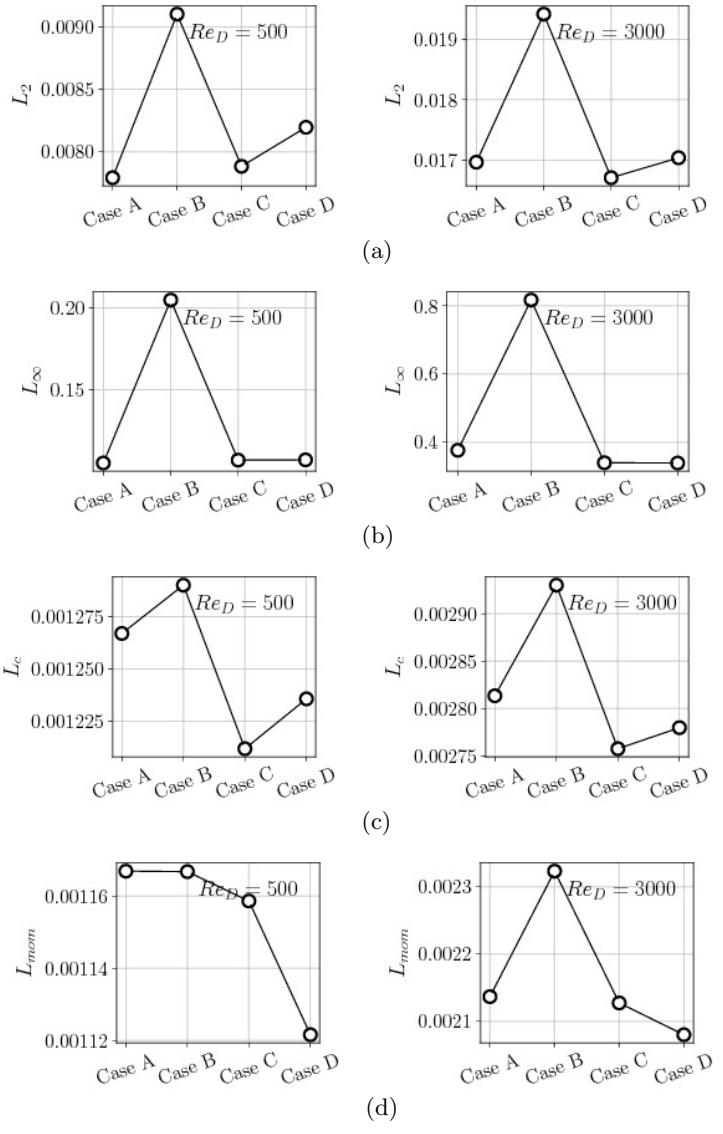


Figure 17: Adversarial training weight dependencies of (a) L_2, (b) L_∞, (c) L_c, and (d) L_{mom} errors from prediction after δt, where $\delta t = 20\Delta t U_\infty / D = 0.01$.

Denton, Emily L, Chintala, Soumith & Fergus, Rob 2015 Deep generative image models using a Laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 1486–1494.

Freymuth, Peter 1966 On transition in a separated laminar boundary layer. *Journal of Fluid Mechanics* **25** (04), 683–704.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron & Bengio, Yoshua 2014 Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680.

Guo, X., Li, W. & Iorio, F. 2016 Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 481–490. ACM.

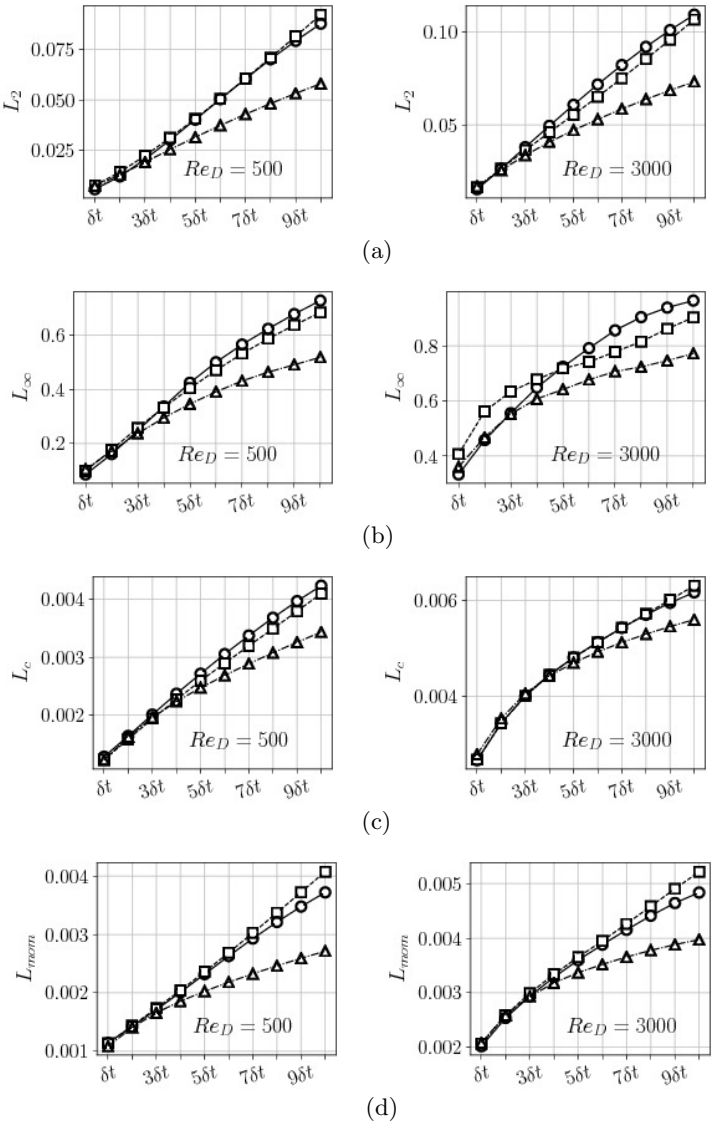


Figure 18: Number set dependencies of (a) L_2, (b) L_∞, (c) L_c, and (d) L_{mom} errors from recursive predictions of Case 1. ∘ and solid line denotes errors from N_{32}; □ and dashed line denotes errors from N_{64}; △ and dash-dotted line denotes errors from N_{128}. $1\delta t = 20\Delta t U_\infty / D = 0.01$.

Kingma, Diederik P & Ba, Jimmy 2014 Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Krizhevsky, Alex, Sutskever, Ilya & Hinton, Geoffrey E 2012 Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105.

Ling, Julia, Kurzawski, Andrew & Templeton, Jeremy 2016 Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics* **807**, 155–166.

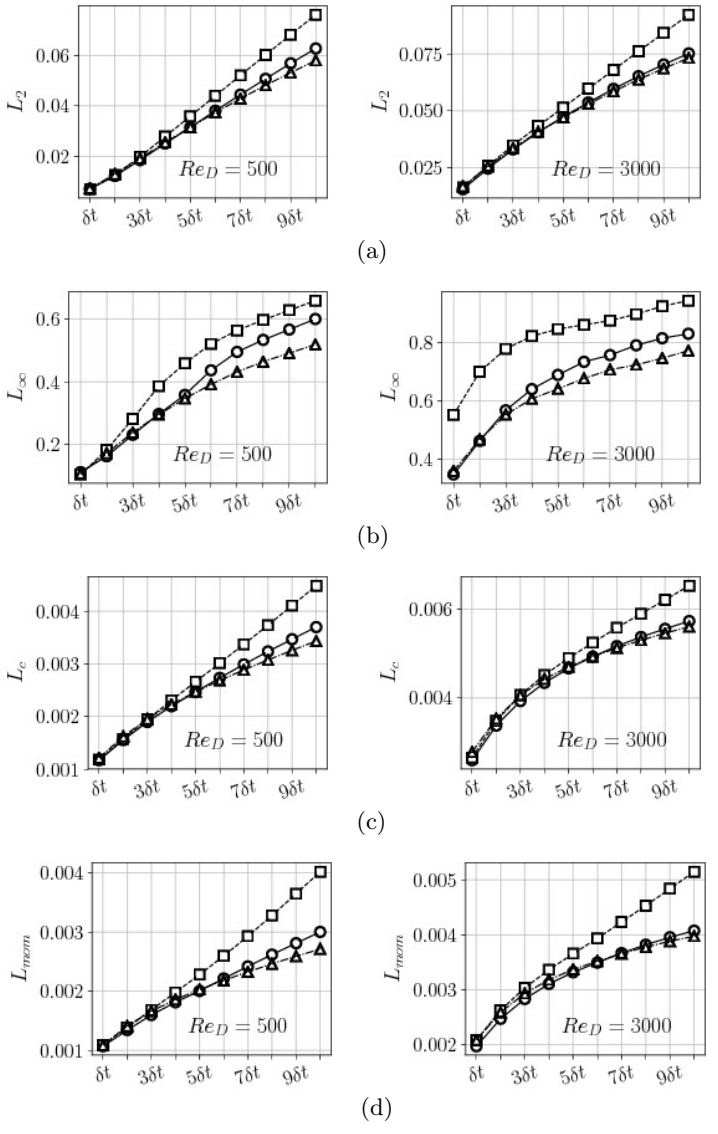


Figure 19: Configuration dependency of the generator model. (a) L_2, (b) L_∞, (c) L_c, and (d) L_{mom} errors from recursive predictions of Case 1. The time-step interval between flow fields is $\delta t = 20\Delta t U_\infty / D = 0.01$. ∘ and solid line denotes errors from GM_{16}; □ and dashed line denotes errors from GM_{18}; △ and dash-dotted line denotes errors from GM_{20}.

Marcus, Philip S 1988 Numerical simulation of Jupiter's great red spot. *Nature* **331** (6158), 693–696.

Mathieu, Michael, Couprie, Camille & LeCun, Yann 2015 Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440* .

Mezić, Igor 2013 Analysis of fluid flows via spectral properties of the Koopman operator. *Annual Review of Fluid Mechanics* **45**, 357–378.

Miyanawala, T. P. & Jaiman, R. K. 2017 An efficient deep learning technique for the

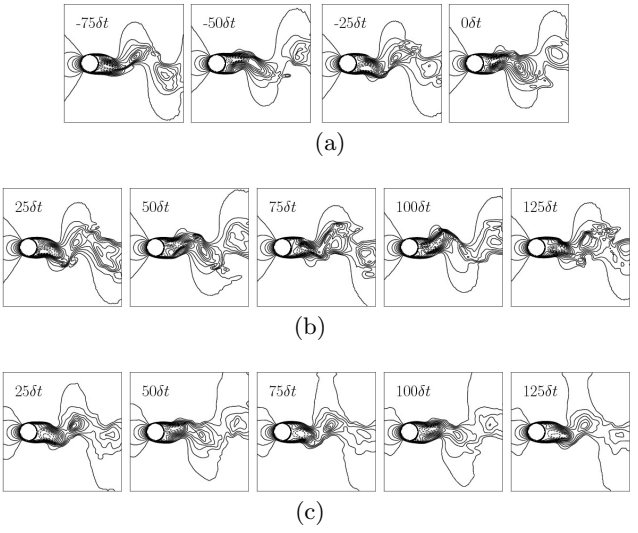


Figure 20: Comparison of the streamwise velocity (u/U_∞) at $Re_D = 500$ after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$, where $1\delta t = 20\Delta t U_\infty / D = 0.01$. Flow fields at $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-row: ground truth; (c)-row: predicted by the GAN without physical loss functions. 15 contour levels from -0.5 to 1.0 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

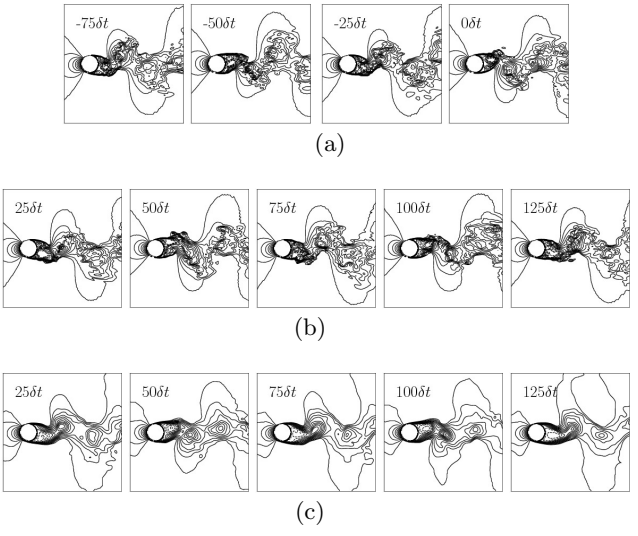


Figure 21: Comparison of the streamwise velocity (u/U_∞) at $Re_D = 3000$ after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$, where $1\delta t = 20\Delta t U_\infty / D = 0.01$. Flow fields at $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-row: ground truth; (c)-row: predicted by the GAN without physical loss functions. 15 contour levels from -0.5 to 1.0 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

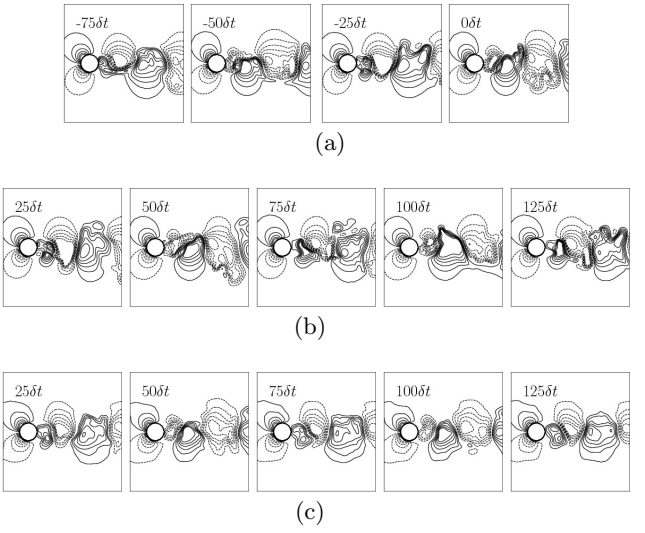


Figure 22: Comparison of the cross-stream velocity (v/U_∞) at $Re_D = 500$ after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-row: ground truth; (c)-row: predicted by the GAN without physical loss functions. 15 contour levels from -0.7 to 0.7 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

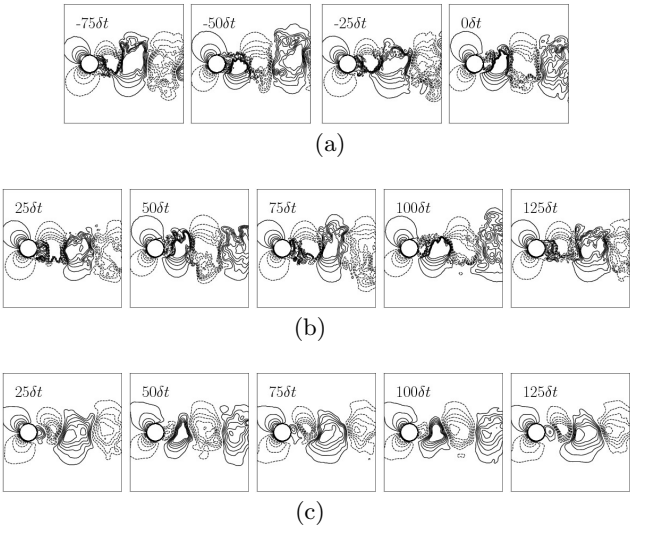


Figure 23: Comparison of the cross-stream velocity (v/U_∞) at $Re_D = 3000$ after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$, where $1\delta t = 20\Delta t U_\infty/D = 0.01$. Flow fields at $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-row: ground truth; (c)-row: predicted by the GAN without physical loss functions. 15 contour levels from -0.7 to 0.7 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

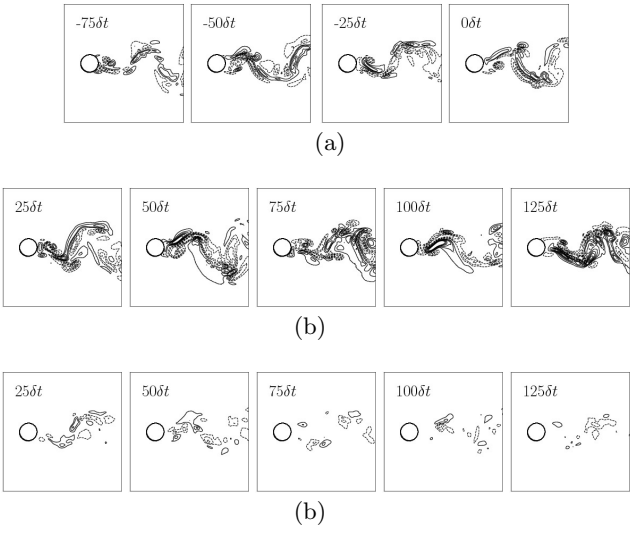


Figure 24: Comparison of the spanwise velocity (w/U_∞) at $Re_D = 500$ after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$, where $1\delta t = 20\Delta t U_\infty / D = 0.01$. Flow fields at $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-row: ground truth; (c)-row: predicted by the GAN without physical loss functions. 15 contour levels from -0.5 to 0.5 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

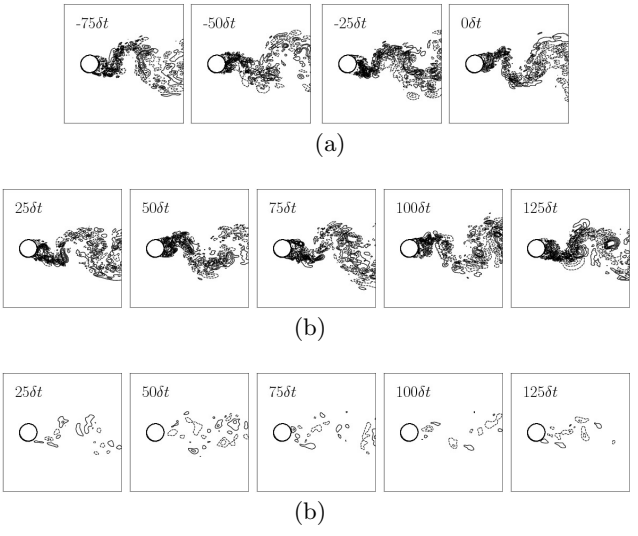


Figure 25: Comparison of the spanwise velocity (w/U_∞) at $Re_D = 3000$ after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$, where $1\delta t = 20\Delta t U_\infty / D = 0.01$. Flow fields at $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-row: ground truth; (c)-row: predicted by the GAN without physical loss functions. 15 contour levels from -0.5 to 0.5 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

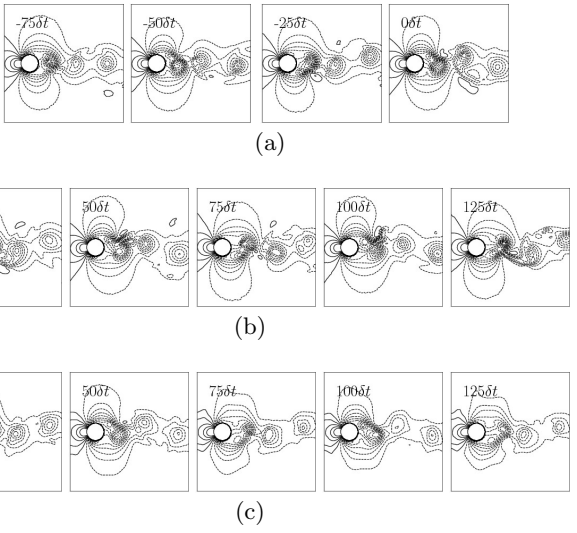


Figure 26: Comparison of the pressure ($p/\rho U_\infty^2$) at $Re_D = 500$ after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$, where $1\delta t = 20\Delta t U_\infty / D = 0.01$. Flow fields at $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-row: ground truth; (c)-row: predicted by the GAN without physical loss functions. 15 contour levels from -1.0 to 0.4 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.

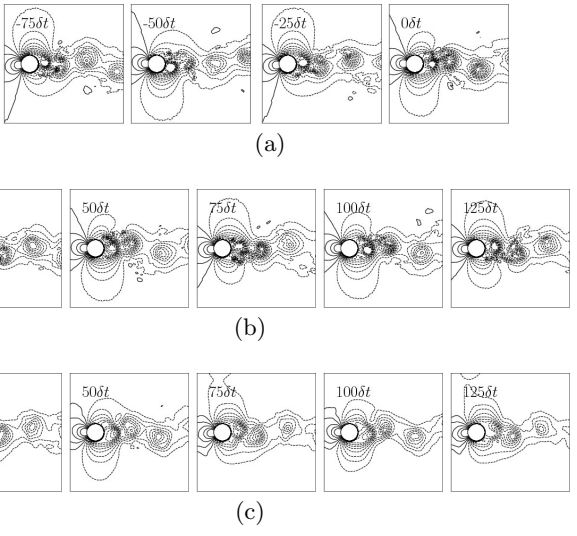
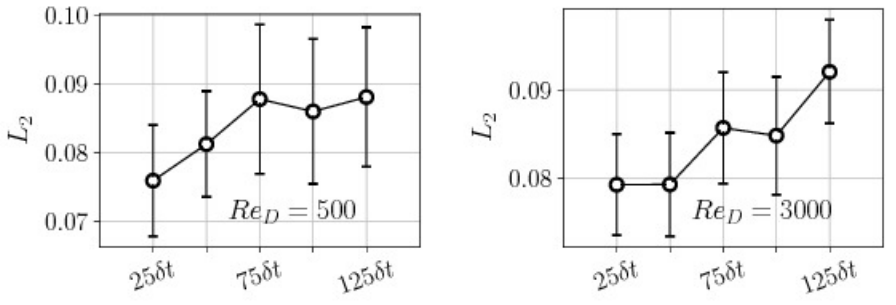


Figure 27: Comparison of the pressure ($p/\rho U_\infty^2$) at $Re_D = 3000$ after $25\delta t$, $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$, where $1\delta t = 20\Delta t U_\infty / D = 0.01$. Flow fields at $50\delta t$, $75\delta t$, $100\delta t$, and $125\delta t$ are recursively predicted (utilizing flow fields predicted prior time-steps as parts of the input). (a)-row: input set; (b)-row: ground truth; (c)-row: predicted by the GAN without physical loss functions. 15 contour levels from -1.0 to 0.4 are shown. Solid line — and dotted line −− indicate positive and negative contour levels, respectively.



(a) (b)

Figure 28: L_2 errors for GAN prediction as a function of δt at (a) $Re_D = 500$ and (b) $Re_D = 3000$. $1\delta t = 20\Delta t U_\infty / D = 0.01$.

Re_D	Flow variable	Time-step interval	Number of recursive steps	L_2 error
500	u/U_∞	δt	25	0.150 ± 0.028
		$25\delta t$	1	0.082 ± 0.011
	v/U_∞	δt	25	0.188 ± 0.015
		$25\delta t$	1	0.110 ± 0.009
	w/U_∞	δt	25	0.065 ± 0.006
		$25\delta t$	1	0.074 ± 0.007
	$p/\rho U_\infty^2$	δt	25	0.143 ± 0.017
		$25\delta t$	1	0.038 ± 0.005
3000	u/U_∞	δt	25	0.162 ± 0.021
		$25\delta t$	1	0.093 ± 0.007
	v/U_∞	δt	25	0.209 ± 0.018
		$25\delta t$	1	0.114 ± 0.0068
	w/U_∞	δt	25	0.069 ± 0.007
		$25\delta t$	1	0.066 ± 0.006
	$p/\rho U_\infty^2$	δt	25	0.121 ± 0.016
		$25\delta t$	1	0.044 ± 0.004

Table 5: Comparison of L_2 errors for each flow variable from predictions using δt time-interval size with 25 recursive steps and using $25\delta t$ time-step interval size with a single step. $1\delta t = 20\Delta t U_\infty / D = 0.01$

Navier-Stokes equations: Application to unsteady wake flow dynamics. *arXiv preprint arXiv:1710.09099* .

van den Oord, Aaron, Kalchbrenner, Nal, Espeholt, Lasse, Vinyals, Oriol & Graves, Alex 2016*a* Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, pp. 4790–4798.

van den Oord, Aaron, Kalchbrenner, Nal & Kavukcuoglu, Koray 2016*b* Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759* .

Radford, Alec, Metz, Luke & Chintala, Soumith 2015 Unsupervised representation

learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

Ranzato, MarcAurelio, Szlam, Arthur, Bruna, Joan, Mathieu, Michael, Collobert, Ronan & Chopra, Sumit 2014 Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604* .

Ruderich, R & Fernholz, HH 1986 An experimental investigation of a turbulent shear flow with separation, reverse flow, and reattachment. *Journal of Fluid Mechanics* **163**, 283–322.

Schmid, Peter J 2010 Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics* **656**, 5–28.

Singh, Anand Pratap, Medida, Shivaji & Duraisamy, Karthik 2017 Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal* pp. 1–13.

Sirovich, Lawrence 1987 Turbulence and the dynamics of coherent structures part I: coherent structures. *Quarterly of Applied Mathematics* **45** (3), 561–571.

Srivastava, Nitish, Mansimov, Elman & Salakhudinov, Ruslan 2015 Unsupervised learning of video representations using LSTMs. In *International Conference on Machine Learning*, pp. 843–852.

Tracey, Brendan, Duraisamy, Karthik & Alonso, Juan 2015 A machine learning strategy to assist turbulence model development. *AIAA Paper* **1287**, 2015.

Wu, Theodore Yaotsu 2011 Fish swimming and bird/insect flight. *Annual Review of Fluid Mechanics* **43**, 25–58.

Wu, Xiaohua & Moin, Parviz 2009 Direct numerical simulation of turbulence in a nominally zero-pressure-gradient flat-plate boundary layer. *Journal of Fluid Mechanics* **630**, 5–41.

Yonehara, Yoshinari, Goto, Yusuke, Yoda, Ken, Watanuki, Yutaka, Young, Lindsay C, Weimerskirch, Henri, Bost, Charles-André & Sato, Katsufumi 2016 Flight paths of seabirds soaring over the ocean surface enable measurement of fine-scale wind speed and direction. *Proceedings of the National Academy of Sciences* **113** (32), 9039–9044.

You, D., Ham, F. & Moin, P. 2008 Discrete conservation principles in large-eddy simulation with application to separation control over an airfoil. *Physics of Fluids* **20** (10), 101515.

Zhang, Ze Jia & Duraisamy, Karthikeyan 2015 Machine learning methods for data-driven turbulence modeling. *AIAA* **2460**, 2015.