Compressed neural networks for reduced order modeling

Yin Zhu (朱银),1,2 Qiangqiang Sun (孙强强),3,a Dandan Xiao (肖旦丹),4,5 Jie Yao (姚杰),2,5 and Xuerui Mao (毛雪瑞)5,6,b

AFFILIATIONS
1 School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 100081, China
2 Advanced Research Institute of Multidisciplinary Sciences, Beijing Institute of Technology, Beijing 100081, China
3 Faculty of Engineering, University of Nottingham, Nottingham NG7 2RD, United Kingdom
4 School of Aerospace Engineering, Beijing Institute of Technology, Beijing 100081, China
5 Beijing Institute of Technology, Zhuhai Beijing Institute of Technology (BIT), Zhuhai 519088, China
6 State Key Laboratory of Explosion Science and Safety Protection, Beijing 100081, China

a) Author to whom correspondence should be addressed: sh1225128@sina.com
b) Electronic mail: maoxuerui@sina.com

ABSTRACT
Reduced order modeling (ROM) techniques, such as proper orthogonal decomposition (POD) and dynamic mode decomposition (DMD), have been widely used to analyze stationary flows. Neural networks such as autoencoders are effective to reduce the dimension of non-stationary flow, but their larger storage requirements compared to POD and DMD compromise the expectations on ROM. The present work aims at compressing the autoencoder model via two distinctively different approaches, i.e., pruning and singular value decomposition (SVD). The developed algorithm is then applied to reconstruct the flow fields of typical stationary (i.e., a laminar cylinder flow and two turbulent channel flows) and non-stationary (i.e., a laminar co-rotating vortex pair) examples. It is shown that pruning and SVD reduce the size of the autoencoder network to 6% and 3% for the two simple laminar cases (or 18% and 13%, 20%, and 10% for the two complex turbulent channel flow cases), respectively, with approximately the same order of accuracy. Therefore, the proposed autoencoders optimized by the network pruning and SVD lead to effective ROM of both stationary and non-stationary flows although they require more iterations to converge than conventional methods.

Published under an exclusive license by AIP Publishing. https://doi.org/10.1063/5.0194598

I. INTRODUCTION

The analysis of unsteady flow fields is challenging due to the massive data and complex dynamics incorporated. As such, various reduced order modeling (ROM) methods have been developed for the dimensionality reduction of the raw flow systems and can be classified into two types: pre-processed ROMs and data-driven ROMs.1,2 The first one is based on the Galerkin projection of the full-state equations into sub-spaces of lower dimensions. The second one, on the other hand, describes the key dynamics of the flow from data without a priori knowledge on the underlying governing equations, e.g., the commonly used dynamic mode decomposition (DMD) or proper orthogonal decomposition (POD).1 More recently, the machine learning (ML) based methods such as deep neural networks were applied for ROM and data forecasting. They overcome some POD/DMD limitations, e.g., the intrinsic difficulty of dealing with physical complexity with a linear superimposition of modes. Hybrid approaches were also reported, using modal decompositions for dimensionality reduction and then deep learning for predictive models based on the obtained lower-dimensional system.1

Among these ROM techniques, DMD and POD have been extensively applied to extract lower-order dominant modes from snapshots of flow fields. For example, Lumley5 was the first one to use the POD method to analyze turbulent flow. The POD mode is orthogonal to each other, and its energy decreases sequentially. Schmid6 proposed the DMD algorithm to find a best-fit linear model describing the time evolution of spatiotemporal coherent structures in data. Overall, these methods are effective to decompose the stationary flow (e.g., flow around a cylinder1) but do not perform well to the non-stationary one16-11 defined as the flow that exhibits spatiotemporal statistical variations.6,12 More specifically, consider a time-periodic sinusoidal traveling wave as an example of the stationary flow, the variation of the flow field across snapshots is localized, or in another view, the
location of the key dynamics does not change with time. Several dominant POD modes can effectively represent the data, and this has been reported as the fast decay of the Kolmogorov n-width in the literature. However, for non-stationary flows, consider an advection-diffusion equation with a Gaussian initial condition as an example, the key features exhibit traveling and diffusing patterns. As a result, the location of the key dynamics changes from snapshot to snapshot, and a large number of POD modes are needed to reconstruct the flow field, which has been described as the slow decay of the Kolmogorov n-width in previous efforts.

Inspired by the lossy image compression techniques, wavelet transform has been explored to extend the scope of ROM to the non-stationary flow. The time-frequency localization property of the wavelet transform introduced in prior studies ensures its capacity to process transient data. However, the applications of modal analysis and reduced-order methods based on wavelet transform have not been pursued actively by the fluid community, despite the fact that these two methods have significantly contributed to image compression.

In recent years, machine learning (ML) has substantially advanced the research in dimensionality reduction and future state prediction. For example, good performances have been reported in feature extraction, dimensionality reduction, super-resolution reconstruction, and prediction. Compared with conventional ROMs, which are mainly used to decompose stationary flow fields into critical modes, ML-based modal analysis methods have no limitation on the stationarity of the flow fields, even though the majority of the existing efforts have been focused on stationary flow fields. Among these methods, the automatic encoder (or autoencoder) has demonstrated outstanding efficacy in nonlinear dimensionality reduction and feature extraction. For example, Murata et al. proposed a novel nonlinear modal decomposition method based on convolutional autoencoders, which significantly improves the reconstruction performances of two-dimensional flow fields compared with the POD method.

While these autoencoder neural networks effectively capture fundamental dynamics, their application to complex flows is restricted due to the substantial storage and computing resource requirements they entail. In addition, neural networks and deep learning models usually suffer from over-parameterization. For example, Denil et al. reported that the parameter redundancy rate of a multilayer perceptron (MLP) neural network employed to reconstruct the Modified National Institute of Standards and Technology (MNIST) database is around 95%. Therefore, the storage demand for the autoencoder-based ROM exceeds those of POD or DMD models, compromising the expectation of achieving a "reduced-order" model.

The network pruning method, as its name implies, can alleviate the storage and computational complexity of the autoencoder network. To compress the network without significantly affecting its performance, Han et al. suggested to learn the essential weight connections in pre-trained models. The HashedNets model developed by Chen et al. accomplishes parameter sharing by grouping the weights into hash buckets using a hash function. Zhou et al. optimized the number of neurons and parameters during the training by adding sparse constraints to the objective function. As a low-order approximation method, singular value decomposition (SVD) is another natural choice to reduce the redundancy of the autoencoder network by constructing a low-rank weight tensor matrix with higher computational efficiency than the original one. For example, Zhang et al. utilized SVD to perform the low-rank decomposition regarding the convolutional neural network and reported that the testing time could be shortened. Lu et al. designed a compact multi-task deep learning framework by decomposing the fully connected layers via the truncated SVD. Similar to pruning techniques, the SVD of autoencoder networks has not been discussed actively in the ROM of flow fields.

In the present work, to test the autoencoder-based ROM performance without redundancy for non-stationary flow, an autoencoder network is applied to model the essential dynamics of non-stationary flow, and its size is further compressed by pruning or SVD. The details of autoencoders, network pruning, low-rank approximation SVD, and POD algorithms are presented in Sec. II, followed by the configuration and data sampling details of four test cases in Sec. III. In Sec. IV, the results of nonlinear dimensionality reduction and reconstruction of the flow field obtained through the conventional and compressed autoencoders are presented. Finally, conclusions are drawn in Sec. V.

II. ALGORITHM

A. Autoencoder

An autoencoder is a specific type of neural network and consists of an encoder and a decoder. To implement feature extraction and dimensionality reduction, the encoder reduces the number of neurons in the hidden layer and maps high-dimensional input samples to a low-dimensional compressed representation. The decoder then maps the encoded representation back to the expected output, which closely resembles the original input. In this work, we focus on the compression and reconstruction of flow fields (e.g., velocity, vorticity, etc.) using the autoencoder, as sketched in Fig. 1. The total number of network layer \( I = 2N - 1 \). The input vector \( \theta \in \Theta \Theta \in \mathbb{R}^{M_1 \times K} \), \( \Theta = \{ \theta_1, ..., \theta_K \} \) represents flow field variables, e.g., velocity, vorticity across the domain, with \( K \) denoting the total number of snapshots and \( M_1 \) the total number of points in each snapshot. The encoding and decoding processes are governed by

\[
\bar{h}_l = \begin{cases} 
\theta & \text{if } l = 1, \\
\Phi_l(W_l \bar{h}_{l-1} + b_l) & \text{if } l = 2, ..., I,
\end{cases}
\]

where \( \bar{h}_l, W_l, b_l, \) and \( \Phi_l \) are the result, weight, bias, and activation function in the \( l \)-th layer of the autoencoder network with number of neurons \( M_l \), respectively. Clearly, \( M_I \) represents the length of the input vector as defined above and by definition \( M_1 = M_I \).

In the training process, \( W_l \) and \( b_l \) are adjusted so that the reconstruction error between \( \theta \) and the output vector \( \bar{h}_I \) is minimized,

\[
\mathcal{L} = \sum_{i=1}^{E} \left\| (\theta - \bar{h}_I) \right\|_2^2 / \sqrt{M_I E},
\]

where \( \mathcal{L} \) is the loss function, \( E \) denotes the batchsize, \( 0 < E \leq K \) represents the batchsize (namely, the number of snapshots used to train the model in a single iteration), \( \left\| \cdot \right\|_2 \) is the Euclidean norm of a vector, and the subscript \( i \) denotes the \( i \)-th component of the batchsize \( E \).

In Secs. II B and II C, two types of network compression techniques (i.e., pruning and SVD) shown in Fig. 2 are discussed.

B. Network pruning

Pruning is a technique for removing edges, neurons, or even layers in neural networks. It has been demonstrated as an effective
method to reduce the computational cost while maintaining accuracy in classification and regression problems. In the pruning method, the network first undergoes a pre-trained phase to identify and learn the essential connections within an autoencoder network. Subsequently, the network is pruned by eliminating connections with weights below a specified threshold, resulting in a sparse network. Finally, the sparse network is retrained to ensure that the model’s performance is only minimally affected. The above steps are repeated iteratively until convergence is achieved. The algorithm uses a randomly initialized pruning rate (defined as the ratio of the number of pruned to the number of all parameters, e.g., 10%) as input. It outputs the weight threshold defined as the critical value determining whether weight connections and neurons need to be pruned and generates a mask matrix $Q_l$ with the same size as the weight matrix $W_l$. The element of $Q_l$ is 0 where the corresponding weight is less than the threshold and 1 elsewhere. The raw weight matrix is then multiplied by the mask matrix to modulate the weights,

$$\tilde{W}_l = W_l \cdot Q_l,$$

(3)

where $\tilde{W}_l$ represents the output weight matrix. Then, the weight matrix $W_l$ in Eq. (1) can be approximated by $\tilde{W}_l$.

C. SVD

Given the fact that the neural network is typically over-parameterized, the weight matrix can be represented using matrices or tensors with lower ranks. For the SVD of the autoencoder network, the weight matrix of the fully connected layer is initially decomposed using the truncated SVD method. The adjusted network is then retrained to improve its accuracy.

SVD decomposes a matrix $W_l \in \mathbb{R}^{M_l \times M_i}$ into three terms

$$W_l = R_l Y_l V_l,$$

(4)

where $Y_l \in \mathbb{R}^{A \times A}$ [$A$ is the rank of $W_l$, and $A \leq \min(M_{i-1}, M_i)$] is a diagonal matrix with non-negative singular values on its diagonal.
arranged in descending order. \( R_i \in \mathbb{R}^{M \times A} \) and \( V_i \in \mathbb{R}^{A \times M_i-1} \) are orthogonal matrices.

By retaining only the most significant \( B (B \leq A) \) singular values of \( W_i \), corresponding to the first \( B \) diagonal entries of matrix \( Y_i \), an approximation matrix is constructed according to the following equation: \(^{34}\)

\[
\tilde{W}_i = R_i Y_i V_i,
\]

where \( R_i \in \mathbb{R}^{M \times B} \), \( Y_i \in \mathbb{R}^{B \times B} \), and \( V_i \in \mathbb{R}^{B \times M_i-1} \) are the truncated matrices of \( R_i \), \( Y_i \), and \( V_i \), respectively.

Substituting Eq. (5) into Eq. (1), one obtains

\[
h_i = \begin{cases} 
\theta & \text{if } l = 1, \\
\Phi_i \left[ R_i \left ( Y_i V_i \right ) h_{l-1} + b_l \right ] & \text{if } l = 2, \ldots, I. 
\end{cases}
\]

In SVD, an original fully connected layer is decomposed into an unbiased \( M_1 \times B \) layer and a \( B \times M_i-1 \) layer with bias \( b_l \). Then, the total number of weights is reduced from \( M_1 \times M_i-1 \) to \( B(M_1 + M_i-1) \), with a compression ratio \( \frac{B(M_1 + M_i-1)}{M_1 \times M_i-1} \), which can be approximated to \( M_1/B \) when \( M_i \ll M_1 \). The encoder or \( M_1/B \) when \( M_i \gg M_1 \) (the decoder). The number of weights can be controlled by modifying the number of singular values retained (i.e., \( B \)). Moreover, it should be noted that this SVD technique is independent of the pruning addressed in Sec. II B, and it acts on and compresses the original full weight matrix.

### D. POD

For completeness, the ROM based on POD method is briefly presented here. The POD, also known as principal component analysis, is a data processing technique widely applied in various engineering areas, including weather forecasting, data compression, stochastic processes, and oceanography. For POD, the input matrix \( \Theta \) defined in Sec. II A can be decomposed into \( K \) orthogonal modes via SVD assuming there are no zero singular values and \( K \ll M_i \),

\[
\Theta = \begin{bmatrix} g_1 & \cdots & g_K \end{bmatrix} \begin{bmatrix} \hat{\lambda}_1 & & \\ & \ddots & \\ & & \hat{\lambda}_K \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_K \end{bmatrix},
\]

where \( \hat{\lambda}_i, g_i, \) and \( h_i \) (\( i = 1, \ldots, K \)) are the amplitude descending with \( i \), the orthogonal spatial mode, and the temporal orthogonal mode, respectively.

Each POD mode is obtained by combining a spatial mode with a temporal one.\(^ {35} \) A ROM representation can be constructed by retaining the first few most energetic modes or singular values.

### III. DATA COLLECTION

An autoencoder network is used to achieve the modal decomposition of the flow fields. The input is the dataset of vorticity or streamwise velocity collected from the following four cases, covering laminar (cases 1 and 2) and turbulent flows (cases 3 and 4) as well as stationary (cases 1, 3, and 4) and non-stationary ones (case 2).

### A. Laminar flow around a cylinder

In this example, a dataset of the vorticity field for flow around a circular cylinder, as shown in Fig. 3, is used. This case has been well employed as a benchmark example for validating the effectiveness of ROM.\(^ {3,2,23,36} \) The two-dimensional incompressible Navier–Stokes equations are solved using a solver based on the immersed boundary projection and fast multi-domain methods.\(^ {37} \) A Cartesian grid is adopted, and the no-slip boundary condition on the cylinder surface is imposed. The Reynolds number, defined as \( Re = DU_\infty/\nu \), where \( D \) is the cylinder diameter, \( U_\infty \) is the free-stream velocity, and \( \nu \) is the kinematic fluid viscosity, is 100. The computational domain, covering \( 72D \times 32D \), is discretized into a grid with 450 \times 200 points.\(^ {36} \) The dataset contains a total of 150 snapshots.

### B. Co-rotating vortex pair flow

The second example used to validate the proposed methods is a time-evolving co-rotating vortex pair,\(^ {36} \) as can be seen in Fig. 4. The linear combination of two Lamb–Oseen vortices is used to generate the initial condition, which can be expressed in the two-dimensional Cartesian coordinates as

\[
u(t = 0) = \frac{1}{r_1^2} (y - y_1)[1 - e^{-r_1^2}] - \frac{1}{r_2^2} (y - y_2)[1 - e^{-r_2^2}],
\]

\[
u(t = 0) = \frac{1}{r_1^2} (x - x_1)[1 - e^{-r_1^2}] + \frac{1}{r_2^2} (x - x_2)[1 - e^{-r_2^2}],
\]

where \( u \) and \( v \) are the velocity components in the \( x \) and \( y \) directions, respectively. \( r_1^2 = (x - x_1)^2 + (y - y_1)^2 \), \( r_2^2 = (x - x_2)^2 + (y - y_2)^2 \), \( (x_1, y_1) = (-3, 0) \), and \( (x_2, y_2) = (3, 0) \) are the initial locations of the two individual vortex cores.\(^ {36} \) Here, \( Re \) is defined as \( Re = DU_\infty/\nu \),\(^ {38} \) where \( DU \) is the dimensional velocity excess in the core of each individual vortex, \( R_0 \) is the radius of an individual vortex core at time \( t = 0 \), which is equal to 100. A circular computational domain centered at \( (x, y) = (0, 0) \) with a radius of 100 is discretized to 81 424 mesh cells. A snapshot is collected every five time steps, and a total of 500 snapshots are extracted from the co-rotating vortex pair. One out of every five snapshots is selected as test data, while the remaining 400 snapshots are used as the training set.

### C. Turbulent channel flow

Finally, two turbulent channel flow cases are considered to test the capacity of the proposed method beyond the laminar scope. The direct numerical simulation (DNS) of the Navier–Stokes equations is conducted using the code “POONPACK,” developed by Lee and Moser.\(^ {39} \) The incompressible Navier–Stokes equations are solved using
the method of Kim et al.,40 in which equations for the wall-normal vorticity and the Laplacian of the wall-normal velocity are time-advanced. A Fourier–Galerkin method is used in the streamwise (x) and spanwise (z) directions, while the wall-normal (y) direction is represented using a seventh-order B-spline collocation method. A low-storage implicit–explicit scheme based on third-order Runge–Kutta for the nonlinear terms and Crank–Nicolson for the viscous terms is used for time advance. The flow is driven by a pressure gradient, which varies in time to ensure that the mass flux through the channel remains constant. The friction Reynolds number (Re$_f$ = $u_s h / 
u$, with $u_s = (\tau_w / \rho)^{1/2}$ the friction velocity, $\tau_w$ is the wall shear stress, and $\rho$ is the density) of two test cases is 110 (see Fig. 5) and 150, respectively. The computation domain is set to $2\pi h \times 2h \times \pi h$ and discretized using a Cartesian grid of $N_x \times N_y \times N_z = 64 \times 64 \times 64$ or $96 \times 96 \times 96$ for the Re$_f$ = 110 or 150 case, respectively. For each case, a y–z plane at $x = 2\pi h$ is extracted as the data sampling region. A total of 3781 or 4200 snapshots are employed with a time interval of $\Delta t = h / U_b$ for the Re$_f$ = 110 or 150 case, respectively.

IV. RESULT

A. Hyperparameter analysis

An autoencoder has several hyperparameters, e.g., activation functions, node counts in the bottleneck layer, and the number of layers. Using the flow around a circular cylinder as an example, a trial-and-error approach is employed to assess and optimize these hyperparameters. Specifically, the number of network layer $I$ is set to 5, 7, 9, or 9. The corresponding activation function is configured as tanh, ReLU, or softplus.41

The training performances of the revised autoencoder networks are presented in Fig. 6, showing the validation loss vs the number of epochs for different activation functions. The results reported here are obtained from the autoencoder network with $M_N = 3$ and $I = 5$. 

FIG. 4. Time evolution of the vorticity of the co-rotating vortex pair flow with Re = 100.

FIG. 5. The instantaneous streamwise velocity of turbulent channel flow with Re$_f$ = 110.

FIG. 6. Effects of various activation functions on the validation loss of autoencoder networks for the flow around a cylinder case.
The superior performance of the tanh activation function over the other two alternatives is evident. Consequently, it will be utilized in subsequent investigations.21

Figure 7 shows the effect of the number of nodes $M_N$ in the bottleneck layer on the prediction performance of the autoencoder. Note that the results reported here are obtained with the number of network layers $I = 5$ and the tanh activation function discussed earlier. One can observe that the one-node bottleneck layer aggravates the training performance of the autoencoder network, featuring prolonged oscillation and slow convergence speed. As $M_N$ increases beyond 3, the convergence performance does not exhibit notable improvement.

The reconstructed flow field by the autoencoder network with different numbers of nodes $M_N$ in the bottleneck layer is visualized in Fig. 8. Compared to the reference case shown in Fig. 3, the reconstructed field for the $M_N = 1$ case can only capture the strong vorticity region near the cylinder, and a notable difference is observed in the wake region. Consistent with the observations in Fig. 7, the reconstructed field closely resembles the reference case when $M_N \geq 3$. Thus, $M_N = 3$ is selected for the autoencoder network to avoid over-paramaterization.

Similarly, upon comparing the reconstructed results of the co-rotating vortex pair case shown in Fig. 9 with the reference one illustrated in Fig. 4, the structure of the co-rotating vortex pair cannot be
captured by the autoencoder network with $M_N = 1$. However, the reconstructed difference is reduced significantly when $M_N \geq 3$. Therefore, similar to the first case, the value of $M_N$ is also chosen as 3 for the co-rotating vortex pair case.

However, for two turbulent channel cases, characterized by a much broader range of scales, it is evident that a larger number of modes in the bottleneck layer $M_N$ is required to effectively capture all these scales via comparing the reconstructed flow field shown in Fig. 10 with the reference one shown in Fig. 5. Hence, after balancing the prediction performance and the number of parameters in autoencoder networks, $M_N$ is set as 16 for these two cases.

Finally, the effect of the number of network layers $I$ on the accuracy of the autoencoder is investigated for the cylinder flow case. As shown in Fig. 11, the validation loss for the $I = 5$ case rapidly drops initially and remains almost constant when the epoch is above 25. In addition, as $I$ increases from 5 to 9, the loss remains almost the same after 50 epochs. However, beyond this point, the model experiences overfitting, leading to a degraded performance. Therefore, to ensure optimal model performance and achieve miniaturization of the model size, the number of network layer $I$ is chosen to be 5.

In summary, based on the observations from Figs. 6–8 and 11, the optimized autoencoder network for the cylinder flow has five layers and three nodes in the bottleneck layer and uses tanh as the activation function. Furthermore, the parameter configurations of autoencoder networks constructed for the co-rotating vortex and two turbulent channel flow cases remain identical to those of the cylinder flow case, with the exception that an additional 13 nodes in the bottleneck layer are employed for the last two cases.

Moreover, three train/test ratios, namely, 7:3, 4:1, and 9:1, are tested for the flow around a cylinder case by using the optimized autoencoder network, and their influences on the validation loss of the autoencoder are shown in Fig. 12. It is clear that the results are minorly affected. Therefore, the train/test ratio of 4:1 is adopted in the current work. Specifically, the first 120 consecutive snapshots are used as the training set, while the remains are employed as the testing set.

B. Network compression

In this section, the performances of the previously discussed autoencoder network compression techniques are assessed. Figures 13 and 14 show the validation loss of the autoencoder for the flow around a cylinder case with different train/test ratios.
and 14 show the dependence of validation loss [see Eq. (2)] on the pruning rate and number of features (i.e., singular values) retained via the SVD method, respectively. For the first two test cases, when the network pruning rate is below 92% or the number of retained features is larger than 20, the performance of the network is rarely affected with the corresponding loss being less than 0.001. A comparable level of reconstruction accuracy can be achieved for the turbulent channel flow case when the network pruning rate is less than 80% and the number of retained features is larger than 175. It is worth mentioning that only a small proportion of weights has significant magnitudes and contributions for the reconstruction for the first two flow cases. In contrast, the contribution of weights for reconstruction is distributed more uniformly across the entire network for the turbulent flow case. Therefore, the pruning rate for the turbulent channel flow is lower than that for the other two cases to achieve the same level of accuracy.

Then, a compressed autoencoder network based on the aforementioned criteria is deployed to reconstruct the essential dynamics of stationary and non-stationary flows. The predicted vorticity and the corresponding absolute difference defined as \( \| \theta - \hat{\theta} \| \) for the cylinder flow case are visualized in Fig. 15. For comparison, the reconstruction using the traditional POD method is also provided. It is clear that using the uncompressed autoencoder, the reconstructed flow field shows good agreement with the reference data (Fig. 3). In addition, both the pruning and SVD compressed autoencoders produce reasonably good results for reconstruction, with only minor errors in the wake region, particularly in the case of pruning. Regarding POD with three modes, it also effectively reconstructs the flow.

The reconstruction errors defined as \( \| \theta - \hat{\theta} \| \) of the conventional autoencoder, autoencoders optimized by pruning and SVD, and POD with three modes are 0.68%, 0.80%, 1.04%, and 1.51%, respectively. Therefore, for such a stationary flow, both POD and...
autoencoder achieve good reconstructions by retaining a few key features, i.e., three modes for POD and three nodes in the bottleneck for autoencoder. Furthermore, the significantly compressed autoencoders reach almost the same level of accuracy as the uncompressed one.

For the co-rotating vortex pair case, the visualization of the vorticity reconstruction at time $t = 40$ and its corresponding difference are shown in Fig. 16. For this non-stationary flow, POD with only three modes fails to capture the essential features of the flow, and notable differences, both in the center and tail regions, are observed. However, both the uncompressed and compressed autoencoders can well reproduce the DNS results, even in the thin tail region. The reconstruction errors of the original autoencoder, pruning-based and SVD-based autoencoders and POD with three modes are 3.70%, 6.56%, 9.95%, and 51.97%, respectively. For such non-stationary flows, POD method is outperformed by autoencoders when retaining the same number of modes or features. This can be attributed to the nonlinear activation function in the machine learning model, which enables the incorporation of more information on the dynamics, apart from that POD is naturally ineffective to non-stationary flows.

Figure 17 shows the visualization of the reconstructed streamwise velocity and its corresponding difference for the turbulent channel flow case at $Re = 110$. The corresponding reconstruction errors are 1.94%, 5.48%, 4.74%, and 13.11% for the original autoencoder, pruning-based and SVD-based autoencoders, and POD methods, respectively. It is striking that autoencoders with/without compression show comparable reconstruction performance, not only in the
near-wall region but also in the channel center. However, even with 16 modes, POD fails to model such flows accurately. Intuitively, the distribution of the reconstruction error for POD is more regular. For example, notable reconstruction errors with coherent structures are mainly concentrated in the wake region of the laminar cylinder flow (Fig. 15), vortex merging region of the co-rotating vortex pair (Fig. 16), and the near-wall region of the turbulent channel flow cases (Fig. 17). This is mainly caused by two inherent limitations of POD: (1) it captures the most energetic structures rather than the large-scale ones; (2) it is a linear method becoming ineffective in the region with strong nonlinear characteristics. On the contrary, the autoencoder network is a nonlinear method that can effectively extract the nonlinear characteristics of the flow field, resulting in a smaller and random reconstruction error resembling the white noise.

To examine the effectiveness of the developed algorithm on the turbulent channel flow case at higher Reynolds number, the case at $Re_c = 150$ is also reconstructed and visualized in Fig. 18. Compared to the low $Re_c$ case shown in Fig. 17, the reconstruction differences of the raw/compressed autoencoder increase dramatically. This is caused by the fact that the fully connected neural network cannot capture the multi-scale and nonlinear characteristics of flow fields with the increasing of $Re_c$, and the convolutional neural network autoencoders may be employed to overcome the challenge in future work.

The root mean square error (RMSE), reconstruction error, and compression performances of autoencoder networks with/without modifications are summarized in Table I. For the four flow cases, the RMSE of the reconstructed quantities predicted by the original autoencoder is 0.32%, 0.28%, 1.24%, and 11.88%, respectively. Also, the compression rates of the raw autoencoder networks achieved by pruning/SVD are more than 93% and 79% for the first and last two cases, respectively. Although additional slight RMSE is caused by the remarkable compression of the network, velocity reconstructed by autoencoders optimized

![Figure 17](image1.png)

**FIG. 17.** The distributions of the reconstructed streamwise velocity and corresponding absolute differences $|\theta - \theta_0|$ of the turbulent channel flow case with $Re_c = 110$.

![Figure 18](image2.png)

**FIG. 18.** The distributions of the reconstructed streamwise velocity and corresponding absolute differences $|\theta - \theta_0|$ of the turbulent channel flow case with $Re_c = 150$. 

The authors would like to acknowledge the support from the National Natural Science Foundation of China (Grant Nos. 92152109 and 12202059).
AUTHOR DECLARATIONS
Conflict of Interest
The authors have no conflicts to disclose.

Author Contributions
Yin Zhu: Writing – original draft (equal). Qiangqiang Sun: Writing – review & editing (equal). Dan Dan Xiao: Writing – review & editing (equal). Jie Yao: Writing – review & editing (equal). Xuereui Mao: Supervision (equal); Writing – review & editing (equal).

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

REFERENCES


T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature,” Geosci. Model Dev. 7, 1247–1250 (2014).