Physics of Fluids

RESEARCH ARTICLE | JANUARY 04 2023

Linear attention coupled Fourier neural operator for simulation of three-dimensional turbulence

Wenhui Peng (彭文辉) ; Zelong Yuan (袁泽龙) ; Zhijie Li (李志杰) ; Jianchun Wang (王建春)

Physics of Fluids 35, 015106 (2023)
https://doi.org/10.1063/5.0130334

Physics of Fluids ARTICLE scitation.org/journal/phf

Linear attention coupled Fourier neural operator for simulation of three-dimensional turbulence

Cite as: Phys. Fluids **35**, 015106 (2023); doi: 10.1063/5.0130334
Submitted: 11 October 2022 · Accepted: 9 December 2022 ·
Published Online: 4 January 2023

Wenhui Peng (彭文辉),[1,2,3] Zelong Yuan (袁泽龙),[1,2] Zhijie Li (李志杰),[1,2] and Jianchun Wang (王建春)[1,2,a)]

AFFILIATIONS

[1]Department of Mechanics and Aerospace Engineering, Southern University of Science and Technology, Shenzhen 518055, China
[2]Guangdong-Hong Kong-Macao Joint Laboratory for Data-Driven Fluid Mechanics and Engineering Applications, Southern University of Science and Technology, Shenzhen 518055, China
[3]Department of Computer Engineering, Polytechnique Montreal, Québec H3T 1J4, Canada

[a)]Author to whom correspondence should be addressed: wangjc@sustech.edu.cn

ABSTRACT

Modeling three-dimensional (3D) turbulence by neural networks is difficult because 3D turbulence is highly nonlinear with high degrees of freedom and the corresponding simulation is memory-intensive. Recently, the attention mechanism has been shown as a promising approach to boost the performance of neural networks on turbulence simulation. However, the standard self-attention mechanism uses $O(n^2)$ time and space with respect to input dimension n, and such quadratic complexity has become the main bottleneck for attention to be applied on 3D turbulence simulation. In this work, we resolve this issue with the concept of a linear attention network. The linear attention approximates the standard attention by adding two linear projections, reducing the overall self-attention complexity from $O(n^2)$ to $O(n)$ in both time and space. The linear attention coupled Fourier neural operator (LAFNO) is developed for the simulation of 3D isotropic turbulence and free shear turbulence. Numerical simulations show that the linear attention mechanism provides 40% error reduction at the same level of computational cost, and LAFNO can accurately reconstruct a variety of statistics and instantaneous spatial structures of 3D turbulence. The linear attention method would be helpful for the improvement of neural network models of 3D nonlinear problems involving high-dimensional data in other scientific domains.

I. INTRODUCTION

With the rising of deep learning techniques, neural networks (NNs) have extensively been explored to complement or accelerate the traditional computational fluid dynamics (CFD) modeling of turbulent flows.[1,2] Applications of deep learning and machine learning techniques to CFD include approaches to the improvements of Reynolds averaged Navier–Stokes (RANS) and large eddy simulation (LES) methods. These efforts have mainly focused on using NNs to learn closures of Reynolds stress and subgrid-scale (SGS) stress and, thus, improve the accuracy of turbulence modeling.[3–5]

Deep neural networks (DNNs) have achieved impressive performance in approximating the highly non-linear functions.[6] Guan *et al.* proposed the convolutional neural network model to predict the SGS forcing terms in two-dimensional decaying turbulence.[7] Yang *et al.* incorporated the vertically integrated thin-boundary-layer equations into the model inputs to enhance the extrapolation capabilities of neural networks for large-eddy-simulation wall modeling.[8] Some recent works aim to approximate the entire Navier–Stokes equations by deep neural networks.[9–17] Once trained, the "black-box" NN models can make inference within seconds on modern computers, thus, can be extremely efficient compared with traditional CFD approaches.[18] Xu *et al.* employed the physics-informed deep learning by treating the governing equations as a parameterized constraint to reconstruct the missing flow dynamics.[19] Wang *et al.* further applied the physical constraints into the design of neural network, and proposed a grounded in principled physics model: the turbulent-flow network (TF-Net). The architecture of TF-Net contains trainable spectral filters in a coupled model of Reynolds-averaged Navier–Stokes simulation and large eddy simulation, followed by a specialized U-net for prediction. The TF-Net offers the flexibility of the learned representations and achieves state-of-the-art prediction accuracy.[20]

Most neural network architectures aim to learn the mappings between finite-dimensional Euclidean spaces. They are good at learning a single instance of the governing equation, but they cannot generalize well once the given equation parameters or boundary conditions change.[21–24] Li *et al.* proposed the Fourier neural operators (FNOs),

which learns an entire family of partial differential equations (PDEs) instead of a single equation.[25] The FNO mimics the pseudo-spectral methods;[26,27] it parameterizes the integral kernel in the Fourier space, thus, directly learns the mapping from any functional parametric dependence to the solution.[25] Benefited from the expressive and efficient architecture, the FNO outperforms the previous state-of-the-art neural network models, including U-Net,[28] TF-Net,[20] and ResNet.[29] The FNO achieves 1% error rate on prediction task of two-dimensional (2D) turbulence at low Reynolds numbers.

Direct numerical simulation (DNS) of the three-dimensional turbulent flows is memory intensive and computational expensive, due to the highly nonlinear characteristics of turbulence associated with the large number of degrees of freedom. In recent years, there has been extensive works dealing with the spatiotemporal reconstruction of two-dimensional turbulent flows.[28,30–39] These works reduce the reconstruction error mainly through adopting advanced neural network models[25,35,40–44] or incorporating the prior physical knowledge into the model.[20,21,27,45–48]

However, modeling of three-dimensional turbulence with deep neural networks is more challenging. The size and dimension of simulation data increases dramatically from 2D to 3D.[49,50] In addition, modeling the non-linear interactions of such high-dimensional data requires sufficient model complexity and huge number of parameters with hundreds of layers not being uncommon.[51] Training such models can be computationally expensive because of the sheer amount of parameters involved. Furthermore, these models also take up a lot of memory which can be a major concern during training, since deep neural networks are typically trained on graphical processing units (GPUs), where the available memory is often constrained.

Arvind *et al.* first designed and evaluated two NN models for 3D homogeneous isotropic turbulence simulation.[52] In their work, they proposed two deep learning models: the convolutional generative adversarial network (C-GAN) and the compressed convolutional long-short-term-memory (CC-LSTM) network. They evaluated the reconstruction quality and computational efficiency of the two different approaches. They employed convolutional layers in GANs (CGANs) to handle the high dimensional 3D turbulence data. The proposed CGANs model consists of an eight-layer discriminator and a five-layer generator. The generator takes a latent vector that is sampled from the uniform distribution as an input and produces a cubic snapshot (of the same dimensions as the input) as an output.[52] The CGANs model has an acceptable accuracy in modeling the velocity features of individual snapshots of the flow but has difficulties in modeling the probability density functions (PDFs) of the passive scalars advected with the velocity.[52]

Another model adopts a convolutional LSTM (ConvLSTM) network, which embeds the convolution kernels in a LSTM network to simultaneously model the spatial and temporal features of the turbulence data.[53] However, the major limitation of ConvLSTM is the huge memory cost due to the complexity of embedding a convolutional kernel in an LSTM and unrolling the network,[53] especially when dealing with the high-dimensional turbulence data. The authors resolved this challenge of large-size data memory by training the ConvLSTM on the low dimensional representation (LDR) of turbulence data. They used a convolutional autoencoder (CAE) to learn compressed, low dimensional "latent space" representations for each snapshot of the turbulent flow. The CAE contains multiple convolutional layers, greatly reducing the dimensionality of the data by utilizing the convolutional operators.[52] The convolution filters are chosen since they can capture the complex spatial correlations and also reduce the number of weights due to the parameter-sharing mechanism.[6] The ConvLSTM takes the compressed low dimensional representations as input and predicts future instantaneous flow in latent space which is then "decompressed" to recover the original dimension.[52] The CC-LSTM is able to predict the spatiotemporal dynamics of flow: the model can accurately predict the large scale kinetic energy spectra, but diverges in the small scale range.

Nakamura *et al.* applied the CC-LSTM framework to three-dimensional channel flow prediction task.[54] Despite that the convolutional autoencoder (CAE) can accurately reconstruct the three-dimensional DNS data through the compressed latent space, the LSTM network fails to accurately predict the future instantaneous flow fields.[54] Accurate prediction of three-dimensional turbulence is still one of the most challenging problems for neural networks.

In recent years, attention mechanism has been widely used in boosting the performance of neural networks on a variety of tasks, ranging from nature language processing to computer vision.[55–57] The fluid dynamics community has no exception. Wu *et al.* introduced the self-attention into a convolution auto-encoder to extract temporal feature relationships from high-fidelity numerical solutions.[58] The self-attention module was coupled with the convolutional neural network to enhance the non-local information perception ability of the network and improve the feature extraction ability of the network. They showed that the self-attention based convolutional auto-encoder reduces the prediction error by 42.9%, compared with the original convolutional auto-encoder.[58] Deo *et al.* proposed an attention-based convolutional recurrent autoencoder to model the phenomena of wave propagation. They showed that the attention-based sequence-to-sequence network can encode the input sequence and predict for multiple time steps in the future. They also demonstrated that attention based sequence-to-sequence network increases the time-horizon of prediction by five times compared to the plain sequence-to-sequence model.[59] Liu *et al.* used a graph attention neural network to simulate the 2D flow around a cylinder. They showed that the multi-head attention mechanism can significantly improve the prediction accuracy for dynamic flow fields.[60] Kissas *et al.* coupled the attention mechanism with the neural operators toward learning the partial differential equations task. They demonstrated that the attention mechanism provides more robustness against noisy data and smaller spread of errors over testing data.[61] Peng *et al.* proposed to model the nonequilibrium feature of turbulence with the self-attention mechanism.[40] They coupled the self-attention module with the Fourier neural operator for the 2D turbulence simulation task. They reported that the attention mechanism provided 40% prediction error reduction compared with the original Fourier neural operator model.[40]

The attention mechanism has shown itself to be very successful at boosting the neural networks performance for turbulence simulations, and therefore bringing new opportunities to improve the prediction accuracy of 3*D* turbulence simulation. However, extending the attention mechanism to 3*D* turbulence simulation is a non-trivial task. The challenge comes from the computational expense of the self-attention matrix: the standard self-attention mechanism uses $O(n^2)$ time and space with respect to input dimension n.[55] On the other hand, neural networks are often trained on GPUs, where the memory

is constrained. Such quadratic complexity has become the main bottleneck for the attention mechanism to be extended to 3D turbulence simulations. Detailed computational cost and memory consumption are discussed in Sec. III A.

Recently, Wang *et al.* demonstrated that the self-attention mechanism can be approximated by a low-rank matrix.[62] They proposed the linear attention approximation, which reduces the overall self-attention complexity from $O(n^2)$ to $O(n)$ in both time and space.[62] The linear attention approximation performs on par with standard self-attention, while being much more memory and time efficient,[62] allowing attention module to be applied on high-dimensional data. In this work, we couple the linear attention module with the Fourier neural operator, for the 3*D* turbulence simulation task.

This paper is organized as follow: Sec. II briefly introduces the Fourier neural operator. Section III compares the standard self-attention with linear attention approximation, and introduces the detailed implementation of coupling attention with Fourier neural operator. Section IV describes the details for generating 3*D* turbulence data. Section V discusses the key hyperparamters of the neural network models, including the number of input time steps and the project dimension. In Sec. VI, we benchmark the prediction performance of the original Fourier neural operator vs the linear attention coupled Fourier neural operator, via statistical and physics-based metrics. In Secs. VII and VIII, we provide discussions and draw conclusions, respectively.

II. THE FOURIER NEURAL OPERATOR

The Fourier neural operators learn a mapping between two infinite dimensional spaces from a finite collection of observed input-output pairs. Denote $D \subset \mathbb{R}^d$ as a bounded open set and $\mathcal{A} = \mathcal{A}(D; \mathbb{R}^{d_a}), \mathcal{U} = \mathcal{U}(D; \mathbb{R}^{d_u})$ as separable Banach spaces of function taking values in $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_u}$, respectively.[63] The Fourier neural operators learns an approximation of $\mathcal{A} \to \mathcal{U}$ by constructing a mapping parameterized by $\theta \in \Theta$. The optimal parameters $\hat{\theta} \in \Theta$ are determined in the test-train setting by using a data-driven empirical approximation.[64] The neural operators are formulated as an iterative architecture $v_0 \mapsto v_1 \mapsto \ldots \mapsto v_T$, where v_j for $j = 0, 1, \ldots, T-1$ is a sequence of functions each taking values in $\mathbb{R}^{d_v}$.[18] The FNO architecture, as shown in Fig. 1, consists of three main steps.

First, the input $a \in \mathcal{A}$ is lifted to a higher dimensional representation $v_0(x) = P(a(x))$ by the local transformation P. The local transformation $P : \mathbb{R}^{d_a} \to \mathbb{R}^{d_v}$ acts independently on each spatial component $a(x) \in \mathbb{R}^{d_a}$ of the function $a \in \mathcal{A}$. P is parameterized by a shallow fully connected neural network.

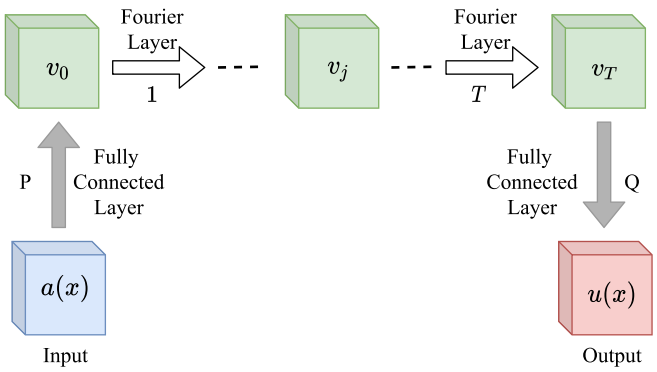

FIG. 1. Fourier neural operator (FNO) architecture.

Then, the higher dimensional representation $v_0(x)$ is updated iteratively. In each iteration, the update $v_t \mapsto v_{t+1}$ is defined as the composition of a non-local integral operator $\mathcal{K}$ and a local, nonlinear activation function σ. The iteration is described by Eq. (1), where $\mathcal{K} : \mathcal{A} \times \Theta_{\mathcal{K}} \to \mathcal{L}(\mathcal{U}(D; \mathbb{R}^{d_v}), \mathcal{U}(D; \mathbb{R}^{d_v}))$ maps to bounded linear operators on $\mathcal{U}(D; \mathbb{R}^{d_v})$ and is parameterized by $\phi \in \Theta_{\mathcal{K}}$. Here, $W : \mathbb{R}^{d_v} \to \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \to \mathbb{R}$ is an elementally defined non-linear activation function,

$$v_{t+1}(x) = \sigma(W v_t(x) + (\mathcal{K}(a; \phi) v_t)(x)), \quad \forall x \in D. \tag{1}$$

Finally, the output $u \in \mathcal{U}$ is obtained by applying the local transformation $u(x) = Q(v_T(x))$, where $Q : \mathbb{R}^{d_v} \to \mathbb{R}^{d_u}$ is parameterized by a fully connected layer.

Let $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse transform of a function $f : D \to \mathbb{R}^{d_v}$, respectively. Replacing the kernel integral operator in Eq. (1) by a convolution operator defined in Fourier space, and applying the convolution theorem, the Fourier integral operator can be expressed by Eq. (2), where R_ϕ is the Fourier transform of a periodic function $\kappa : \bar{D} \to \mathbb{R}^{d_v \times d_v}$ parameterized by $\phi \in \Theta_{\mathcal{K}}$,

$$(\mathcal{K}(\phi) v_t)(x) = \mathcal{F}^{-1}\big(R_\phi \cdot (\mathcal{F} v_t)\big)(x), \quad \forall x \in D. \tag{2}$$

The frequency mode $k \in D$ is assumed to be periodic, and it allows a Fourier series expansion, which expresses as the discrete modes $k \in \mathbb{Z}^d$. The finite-dimensional parameterization is implemented by truncating the Fourier series at a maximal number of modes $k_{\max} = |Z_{k_{\max}}| = |\{k \in \mathbb{Z}^d : |k_j| \le k_{\max,j}, \text{ for } j = 1, \ldots, d\}|$. We discretize the domain D with $n \in \mathbb{N}$ points, where $v_t \in \mathbb{R}^{n \times d_v}$ and $\mathcal{F}(v_t) \in \mathbb{C}^{n \times d_v}$. R_ϕ is parameterized as complex-valued weight tensor containing a collection of truncated Fourier modes $R_\phi \in \mathbb{C}^{k_{\max} \times d_v \times d_v}$, and $\mathcal{F}(v_t) \in \mathbb{C}^{k_{\max} \times d_v}$ is obtained by truncating the higher modes. Therefore, $(R_\phi \cdot (\mathcal{F} v_t))_{k,l} = \sum_{j=1}^{d_v} R_{\phi k,l,j} (\mathcal{F} v_t)_{k,j}$, $k = 1, \ldots, k_{\max}, \quad j = 1, \ldots, d_v$.

In CFD modeling, the flow is typically uniformly discretized with resolution $s_1 \times \cdots \times s_d = n$, and $\mathcal{F}$ can be replaced by the fast Fourier transform (FFT). For $f \in \mathbb{R}^{n \times d_v}, k = (k_1, \ldots, k_d) \in \mathbb{Z}_{s_1} \times \cdots \times \mathbb{Z}_{s_d}$, and $x = (x_1, \ldots, x_d) \in D$, the FFT $\hat{\mathcal{F}}$ and its inverse $\hat{\mathcal{F}}^{-1}$ are given by Eq. (3), for $l = 1, \ldots, d_v$,

$$\begin{aligned} (\hat{\mathcal{F}} f)_l(k) &= \sum_{x_1=0}^{s_1-1} \cdots \sum_{x_d=0}^{s_d-1} f_l(x_1, \ldots, x_d) e^{-2i\pi \sum_{j=1}^{d} \frac{x_j k_j}{s_j}}, \\ \left(\hat{\mathcal{F}}^{-1} f\right)_l(x) &= \sum_{k_1=0}^{s_1-1} \cdots \sum_{k_d=0}^{s_d-1} f_l(k_1, \ldots, k_d) e^{2i\pi \sum_{j=1}^{d} \frac{x_j k_j}{s_j}}. \end{aligned} \tag{3}$$

III. ATTENTION ENHANCED NEURAL NETWORK

A. Self-attention coupled FNO

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and

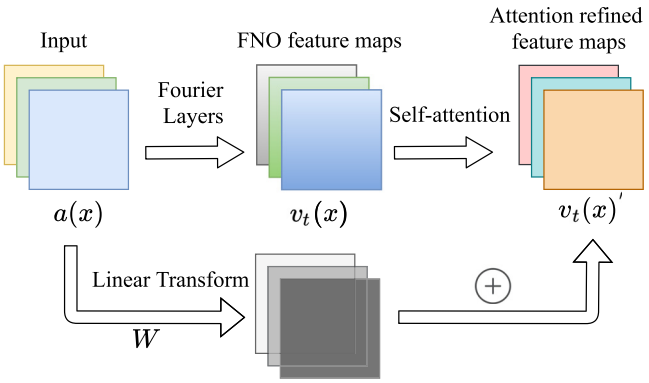

FIG. 2. Attention enhanced Fourier neural operator.

output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.[55] The three sub-modules (query Q, key K, and value V) are the pivotal components of attention mechanism, which come from the concepts of information retrieval systems.[65] Peng *et al.* proposed to couple the attention mechanism with the Fourier neural operator,[40] as shown in Fig. 2. The architecture of the self-attention block is shown in Fig. 3. The convolution parameters W_f, W_g, and W_h learn the embedding of query, key, and value, respectively, and these parameters can be jointly learned with the Fourier layers during training. The self-attention block takes the input tensor of shape $n \times d$, and output attention refined feature maps of the same shape, as shown in Eq. (4). The standard softmax function $\mathbb{R}^K \to (0,1)^K$ is defined by the Eq. (5). In the 2 D turbulence simulation task,[40] the 2 D turbulence data are generated on the grid size of 64×64, such that $n = 64 \times 64$ and $d = 20$,

$$v_t(\boldsymbol{x})' = \text{Attention}(QW_f, KW_g, VW_h) = \underbrace{\text{softmax}\left[\frac{QW_f(KW_g)^T}{\sqrt{d}}\right]}_{P} VW_h, \tag{4}$$

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K. \tag{5}$$

Despite that the self-attention module fits in well with 2D turbulence data, however, extending the self-attention module to 3D turbulence data becomes challenging and difficult. The main reason is that the standard self-attention operation is computationally expensive in both time and space, especially when dealing with long sequence input.[66–68] It is noted from Fig. 3 and Eq. (4) that computing the attention mapping matrix P requires multiplying two $n \times d$ matrices, which incurs a time and space complexity of $O(n^2)$ with respect to input sequence dimension n. In the case of a typical 3D flow field of grid size $64 \times 64 \times 64$, the input sequence dimension m becomes $m = 64 \times 64 \times 64 \times 3$, where the number of physical components is 3, thus computing the attention mapping requires 2034 GB memory for 32-bit floating point data type. Such quadratic complexity on the input sequence dimension has become the main bottleneck for attention to be extended to 3D turbulence data simulations. We resolve this bottleneck with the concept of linear attention.

B. Linear attention approximation

Recently, Wang *et al.* showed that the attention mapping matrix P can be approximated by a low-rank matrix $\tilde{P}$:[62] for any $Q, K, V \in \mathbb{R}^{n \times d}$ and $W_f, W_g, W_h \in \mathbb{R}^{d \times d}$ and for any column vector $w \in \mathbb{R}^n$ of matrix VW_h, there exists a low-rank matrix $\tilde{P} \in \mathbb{R}^{n \times n}$ such that

$$\Pr\left(\|\tilde{P}w^T - Pw^T\| < \epsilon' \|Pw^T\|\right) > 1 - o(1). \tag{6}$$

Here, Pr refers to probability, ϵ' is a small constant, and $o(1)$ is infinitesimal of higher order. Since the attention mapping matrix P is low-rank, it can be approximated by a low-rank matrix P_{low} using the singular value decomposition (SVD) method as below, where σ_i, u_i, and v_i are the i largest singular values and their corresponding singular vectors,

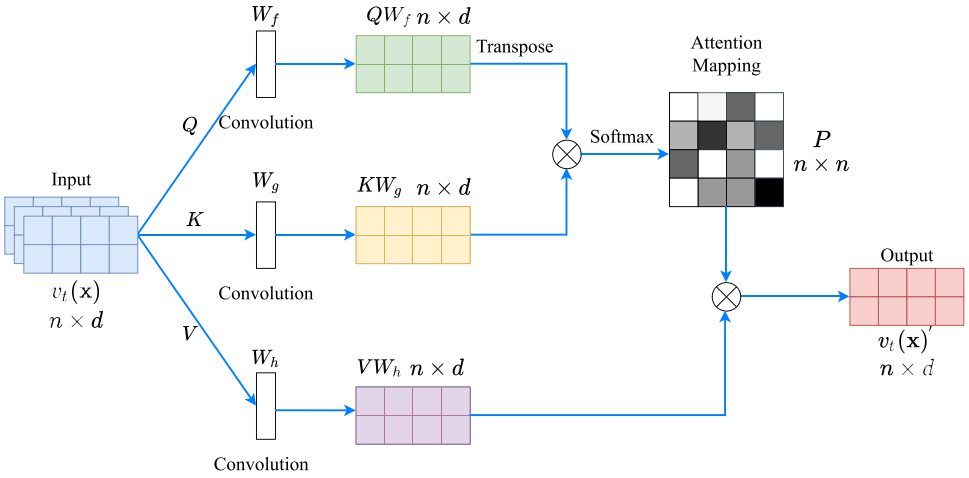

FIG. 3. Standard self-attention module architecture.

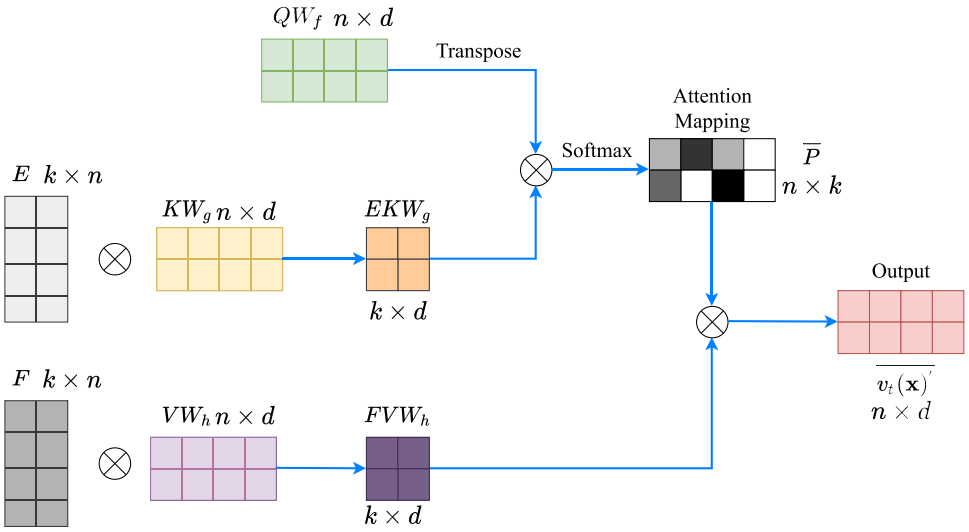

FIG. 4. Linear self-attention module architecture.

$$P \approx P_{\text{low}} = \sum_{i=1}^{k} \sigma_i u_i v_i^T = \underbrace{[u_1, \ldots, u_k]}_{k} \operatorname{diag}\{\sigma_1, \ldots, \sigma_k\} \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} \}k. \quad (7)$$

However, performing an SVD decomposition in the attention mapping matrix P adds additional complexity. Wang *et al.* further proposed an efficient approximation method: the linear self-attention.[62] They showed that the attention output can be approximated by adding two linear projection matrix E and F, based on the distributional Johnson–Lindenstrauss lemma.[62,69]

For any $Q, K, V \in \mathbb{R}^{n\times d}$ and $W_f, W_g, W_h \in \mathbb{R}^{d\times d}$, there exists matrices $E, F \in \mathbb{R}^{n\times k}$ such that,[62] for any row vector w of $QW_f(KW_g)^T/\sqrt{d}$,

$$\Pr\big(\|\text{softmax}(wE^T)FVW_h - \text{softmax}(w)VW_h\| \leq \epsilon'\|\text{softmax}(w)\|\|VW_h\|\big) > 1 - o(1). \quad (8)$$

Detailed mathematical proof can be found in Ref. 62.

The architecture of linear attention is shown in Fig. 4. It projects the original $(n \times d)$-dimensional learned key and value matrix KW_g and VW_h into $(k \times d)$-dimensional projected key and value EKW_g and FVW_h, then computes an (nk)-dimensional attention mapping matrix $\bar{P}$ using the scaled dot-product attention, as described by Eq. (9). The linear projection matrix E and F are not learnable parameters, instead, they are predefined matrix $E = \delta R$ and $F = e^{-\delta}R$, where $R \in \mathbb{R}^{k\times n}$ with i.i.d. entries from Gaussian normal distribution $N(0, 1/k)$ and δ is a small constant.[62]

$$\begin{aligned} \overline{v_t(\boldsymbol{x})}' &= \text{Attention}(QW_f, EKW_g, FVW_h), \\ &= \underbrace{\text{softmax}\left(\frac{QW_f(EKW_g)^T}{\sqrt{d}}\right)}_{\bar{P}:n\times k} \cdot \underbrace{FVW_h}_{k\times d}. \end{aligned} \quad (9)$$

The linear self-attention module performs on par with standard self-attention module,[62] but it offers linear time and memory computation complexity with respect to input sequence length n, since the linear self-attention operations only require $O(nk)$ time and space complexity. For a small projected dimension k where $k \ll n$, the memory and space consumption can be significantly reduced, thus allowing the attention mechanism to be applied on high-dimensional data. In this work, we replace the self-attention block by the linear attention module for 3 D turbulence simulation task. The architecture of our proposed model, the linear attention coupled Fourier neural operator (LAFNO), is shown in Fig. 5. Note that the linear transform W in Fig. 5 is an additional linear transform, different from the linear transform operation inside the Fourier layer,[25] to help training the model.[70] The linear attention reduces the computational cost of computing the attention mapping matrix P, from multiplying two $n \times d$ matrices to multiplying two matrices of $n \times d$ and $k \times d$, where $k \ll n$. In our numerical simulation, the maximum GPU memory consumption of training LAFNO is reduced to 35.82 GB.

IV. DATASET DESCRIPTION

The dimensionless Navier–Stokes equations in conservative form for the 3D incompressible turbulence are given by Pope and Pope[71]

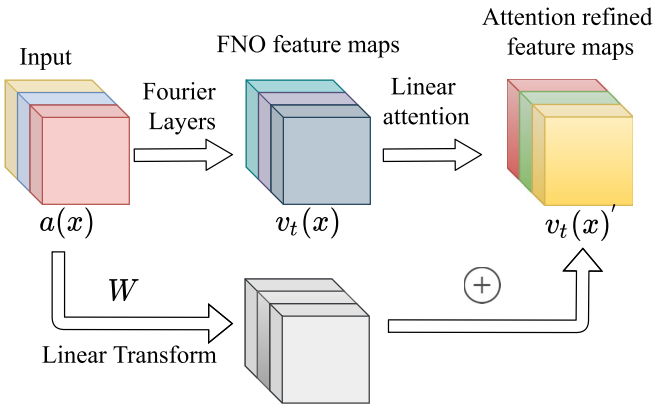

FIG. 5. Linear attention coupled Fourier neural operator (LAFNO).

$$\nabla \cdot \mathbf{u} = 0, \tag{10}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u} + \mathrm{F}, \tag{11}$$

where $\mathbf{u}$ is the velocity, p is the modified pressure divided by the constant density, Re is the Reynolds number, and F is the large-scale force to maintain the turbulence statistically stationary. The initial velocity field $\mathbf{u}(t=0)$ is randomly generated by the Gaussian distribution in spectral space. The initial velocity spectrum of the random velocity field is given by Yuan *et al.*[72]

$$E(k) = A_0 \left(\frac{k}{k_0}\right)^4 \exp\left[-2\left(\frac{k}{k_0}\right)^2\right], \tag{12}$$

where $E(k)$ is the spectrum of kinetic energy per unit mass, k is the wavenumber magnitude in the spectral space. Here, $A_0 = 2.7882$ and $k_0 = 4.5786$.[72] The vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ measures the local rotation of turbulent eddies, and is a Galilean-invariant variable used as the inputs and outputs of the neural networks. Figure 6 shows the vorticity magnitude of ten random initial conditions, which are sampled from the training and testing datasets. Ω is the summation square of vorticity components, as shown in the following equation:

$$\Omega = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}. \tag{13}$$

In this paper, a pseudo-spectral method is applied to numerically simulate the incompressible 3D homogeneous isotropic turbulence in a cubic box of $(2\pi)^3$ on a uniform grid with periodic boundary conditions.[71–73] The velocity can be expanded as the Fourier series,

$$\mathbf{u}(\mathbf{x}, t) = \sum_{\mathbf{k}} \hat{\mathbf{u}}(\mathbf{k}, t) e^{i\mathbf{k}\cdot\mathbf{x}}, \tag{14}$$

where i denotes the imaginary unit, $i^2 = -1$, $\mathbf{k}$ represents the wavenumber vector, $\hat{\mathbf{u}}$ is the velocity in Fourier space, and a hat denotes the variable in wavenumber space. The incompressible Navier–Stokes equations in Fourier space are given by Pope and Pope[71]

$$\mathbf{k} \cdot \hat{\mathbf{u}} = 0, \tag{15}$$

$$\left(\frac{d}{dt} + \frac{1}{\mathrm{Re}} k^2\right) \hat{\mathbf{u}}_j(\mathbf{k}, t) = -ik_l \left(\delta_{jm} - \frac{k_j k_m}{k^2}\right) \sum_{\mathbf{p}+\mathbf{q}=\mathbf{k}} \hat{\mathbf{u}}_l(\mathbf{p}, t) \hat{\mathbf{u}}_m(\mathbf{q}, t) + \hat{\mathcal{F}}_j(\mathbf{k}, t), \tag{16}$$

where $\mathbf{p}$ and $\mathbf{q}$ are the wavenumber vectors and k_j is the jth component of $\mathbf{k}$. The non-local convolution sum at the right-hand side of Eq. (16) is introduced by the nonlinear convection term and is calculated by the pseudospectral method.[71,74] The basic idea is to transform $\hat{\mathbf{u}}_l$ and $\hat{\mathbf{u}}_m$ to $\mathbf{u}_l$ and $\mathbf{u}_m$ in physical space by the inverse fast Fourier transform, and then perform the multiplication in physical space, after that use the forward Fourier transform to determine the convolution sum. The non-local convolution sum calculated by the pseudospectral method can greatly reduce the computational cost but will additionally introduce the aliasing errors. The two-thirds rule is used to eliminate the aliasing errors by truncating the Fourier modes with high wavenumbers.[71,72,74] The large-scale force is constructed by amplifying the velocity field in the wavenumber space to maintain the total kinetic energy spectrum in the first two wavenumber shells to the prescribed values $E_0(1)$ and $E_0(2)$, respectively.[74] The forced velocity $\hat{u}_j^f(\mathbf{k})$ is expressed as

$$\hat{u}_j^f(\mathbf{k}) = \alpha \hat{u}_j(\mathbf{k}), \quad \text{where } \alpha = \begin{cases} \sqrt{E_0(1)/E_k(1)}, & 0.5 \le k \le 1.5, \\ \sqrt{E_0(2)/E_k(2)}, & 1.5 \le k \le 2.5, \\ 1 & \text{otherwise.} \end{cases} \tag{17}$$

Here, $E_0(1) = 1.242\,477$ and $E_0(2) = 0.391\,356$.[74]

Data are generated on a cubic box of $(2\pi)^3$ with uniform grid size of $64 \times 64 \times 64$. Time is advanced with the explicit two-step Adams–Bashforth scheme, where the time step is set to be $\Delta t = 0.002$, and the solution is recorded every $t = 1$ time units.[72] For a partial differential equation $\partial_t \hat{u}_j = \hat{R}_j(\hat{\mathbf{u}}, t)$, the iterative scheme for time advancement is given by Eq. (18), where Δt is the time step,

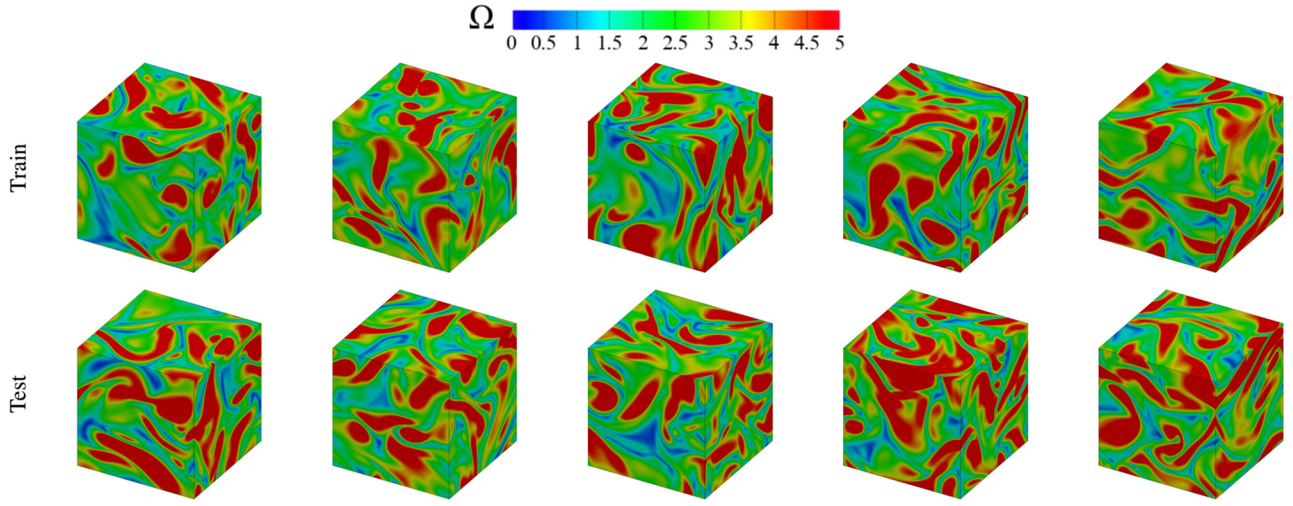

FIG. 6. Ten random sampled initial fields of vorticity magnitude from the training set and testing set.

$t_n = n\Delta t$, and $\hat{u}_j^n$ denotes the velocity in wavenumber space at time t_n. The Taylor Reynolds number of the simulated flow is about 30,

$$\hat{u}_j^{n+1} = \hat{u}_j^n + \Delta t\left[\frac{3}{2}\hat{R}_j(\hat{\mathbf{u}}^n, t_n) - \frac{1}{2}\hat{R}_j\left(\hat{\mathbf{u}}^{n-1}, t_{n-1}\right)\right]. \quad (18)$$

V. HYPERPARAMETERS OF NEURAL NETWORKS

A. Input time steps *T*

The neural network models take a sequence of previous T time steps of flow field as input. The shape of input tensor is $(T \times H \times W \times D \times C)$, where H, W, and D is the height, width, and depth of 3D flow field, respectively, and C is the number of vorticity components. Despite that the velocity field can also be used to train the neural networks, our experiments show that the vorticity field provide slightly better testing prediction accuracy, with 2% error reduction on average. The neural network models predict the flow field of next step as output $(H \times W \times D \times C)$. In the numerical experiments of this paper, $H = W = D = 64$ and $C = 3$. Note that the predicted vorticity at each step is recurrently treated as ground truth and reused as the input (tensors stacked over temporal dimension) with the advance of time. We adopt the standard FNO architecture as described in Ref. 25. It contains four Fourier layers, the number of truncated Fourier modes is 20, and the number of convolution channels is 36.[25]

The number of input time steps T is one of the key hyperparameters. T was set as 10 for the previous study of 2D turbulence simulations.[25,40] However, the input number of ten time steps is too expensive for 3D turbulence simulations, since 3D data are memory-intensive. To make trade-off between the prediction accuracy and memory consuming, we conduct experiments with different number of input time steps T, ranging from 1 to 10, and track the prediction testing error ϵ of next step, as shown in Table I. The relative error ϵ of the model is defined by Eq. (19), where $\tilde{\omega}$ is the predicted vorticity vector and ω is the ground truth.

$$\epsilon = \frac{\|\tilde{\omega} - \omega_2\|}{\|\omega\|_2}, \quad \text{where } \|\mathbf{a}\|_2 = 1/n\sqrt{\sum_{k=1}^{n}|\mathbf{a_k}|^2}. \quad (19)$$

We notice from Table I that the error decreases significantly with the number of input steps from $T = 1$ to $T = 4$, and the error begins to stabilize from $T = 4$. Therefore, we choose $T = 4$ considering both the prediction accuracy and memory consuming.

We fix the number of input time steps to $T = 4$ and investigate the influence of time interval parameter δt on the model accuracy, as shown in Table II. The predicting testing error ϵ is measured at $t = 6$, such that prediction interval spans from $t = 4$ to $t = 6$. We compare four different time intervals $\delta t = 0.2$, 0.4, 1, and 2, corresponding to 100, 200, 500, and 1000 Δt, respectively, where $\Delta t = 0.002$ is the time interval for DNS that is described in Sec. IV. It is noted that the model performs best around $\delta t = 1$. The main reason is that when the time interval δt is too small, the temporal accumulation error increases with the increase in the prediction iterations; meanwhile, when the time interval δt is too large, it becomes difficult for neural networks to capture the temporal correlations of turbulence at different time steps.

TABLE I. Prediction testing errors at different input time steps *T*.

T	1	2	3	4	5	6	7	8	9	10
ϵ	0.401	0.291	0.168	0.095	0.088	0.082	0.086	0.079	0.081	0.085

TABLE II. Prediction testing errors at different time interval δt.

δt	0.2	0.4	1	2
Number of prediction iterations	10	5	2	1
ϵ	0.129	0.122	0.106	0.133

B. Project dimension *k*

Another key hyperparameter is the project dimension k in the linear attention module. $k \ll n$ such that the memory and space can be significantly saved. However, if k is too small, the attention mapping matrix cannot be approximated with sufficient accuracy, which further leads to the increase in the prediction error. In this work, the $3D$ turbulence data are generated on the grid size of $64 \times 64 \times 64$, and the number of convolution channels is set as 36, thus, $n = 64 \times 64 \times 64 \times 3$ and $d = 36$. We aim to find the minimum k that provides maximum memory saving while keeping sufficient accuracy. Table III shows the prediction errors of next step at different k. It is noticed that the prediction error starts to increase when $k < 36$. The main reason is that the attention mapping matrix P is computed by multiplying two $n \times d$ matrices, where $d \ll n$, thus, the attention mapping matrix P is at most rank d and can be approximated accurately for any $k \geq d$.

VI. PERFORMANCE BENCHMARK

Since the prediction errors are produced and accumulated at every step, prediction error increases dramatically with time due to the chaotic features of turbulence. In this section, we evaluate the accumulated prediction errors of two models on temporal dimension: the Fourier neural operator (FNO) and the linear attention coupled Fourier neural operator (LAFNO).

In this numerical experiment, we generate 3000 pairs of input-output data with the numerical solver, where each sample contains 15 steps of solutions of a random initialized condition. The solution is recorded every $t = 1$ time units. Both models (FNO and LAFNO) take the vorticity at previous four time steps solutions as input and give the vorticity at the next time step as output. During training, the vorticity of first four steps $\omega|_{(0,2\pi)^3\times[1,4]}$ is stacked over temporal dimension as the model input, and the model recurrently predicts the vorticity at the next step to fit the vorticity at following 11 steps $\omega|_{(0,2\pi)^3\times[5,15]}$, which are labeled as the ground truth. Specifically, the predicted

TABLE III. Prediction errors at different project dimension *k*.

Project dimension (k)	Error (ϵ)
128	0.0956
72	0.0964
36 ($k = d$)	0.0952
24	0.107
12	0.118

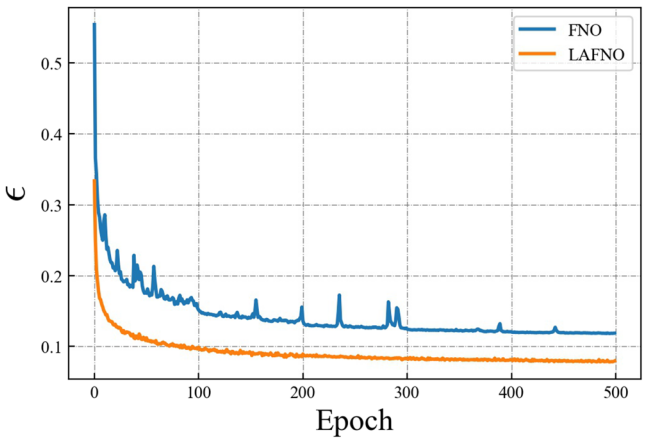

FIG. 7. Learning curve (testing error) of FNO and LAFNO.

vorticity at step 5, $\hat{\omega}|_{(0,2\pi)^3\times[5]}$, is obtained from the input vorticity $\omega|_{(0,2\pi)^3\times[1,4]}$. Then, the predicted vorticity $\hat{\omega}|_{(0,2\pi)^3\times[5]}$ and previous vorticity sequence $\omega|_{(0,2\pi)^3\times[2,4]}$ are stacked over temporal dimension as new input to predict the vorticity at step 6, $\hat{\omega}|_{(0,2\pi)^3\times[6]}$, so on, and so forth. We use 2400 samples for training and 600 samples for testing. The learning curve (testing error) is shown in Fig. 7. It is noted that the LAFNO converges faster and achieves smaller testing error than FNO.

After training, we evaluate both models on the test dataset and compare their performance at three selected time steps $t = 5$, 10, and 15, corresponding to the dimensionless time $t/\tau = 5$, 10, and 15, respectively. Here, $\tau \equiv L_I/u^{rms} = 0.997$ denotes the large-eddy turnover time.[72] L_I denotes the integral scale and u^{rms} is the root mean square (RMS) of velocity. The interval between every two adjacent time steps is $\Delta T = 1$ and the dimensionless time is $\Delta T/\tau = 1$.

Figure 8 visualizes the spatial structures of predicted vorticity field and the relative errors of a test sample. Both models can accurately reconstruct the instantaneous spatial structures of turbulence in the beginning. However, the difference is enlarged significantly as time progresses. The FNO error increases significantly with time, in contrast, the errors of LAFNO are visibly smaller in terms of the region.

Figure 9 shows a 2*D* vorticity slice in the middle of *Z* axis from the same test sample in Fig. 8. At $t/\tau = 5$, both models can accurately reconstruct the instantaneous spatial structures of turbulence. As time progresses to $t/\tau = 10$, the difference can be visibly noticed: the LAFNO can still make accurate reconstructions whereas the FNO cannot, and LAFNO captures the small-scale structures better than FNO. At $t/\tau = 15$, both models fail to reconstruct the instantaneous small-scale structures, whereas the LAFNO can still make relatively accurate reconstructions on the large-scale structures.

Figure 10 shows the spatial-averaged relative errors of the two models with respect to consequent time steps. Both models can make accurate predictions in the beginning ($t/\tau = 5$) with about 10% error. However, since the predictions at each step is recurrently treated as ground truth and reused as the inputs with the advance of time, the prediction errors of both models are accumulated iteratively. As time progresses from $t/\tau = 5$ to $t/\tau = 15$, the error standard deviation becomes larger, and the mean errors of FNO and LAFNO increase to

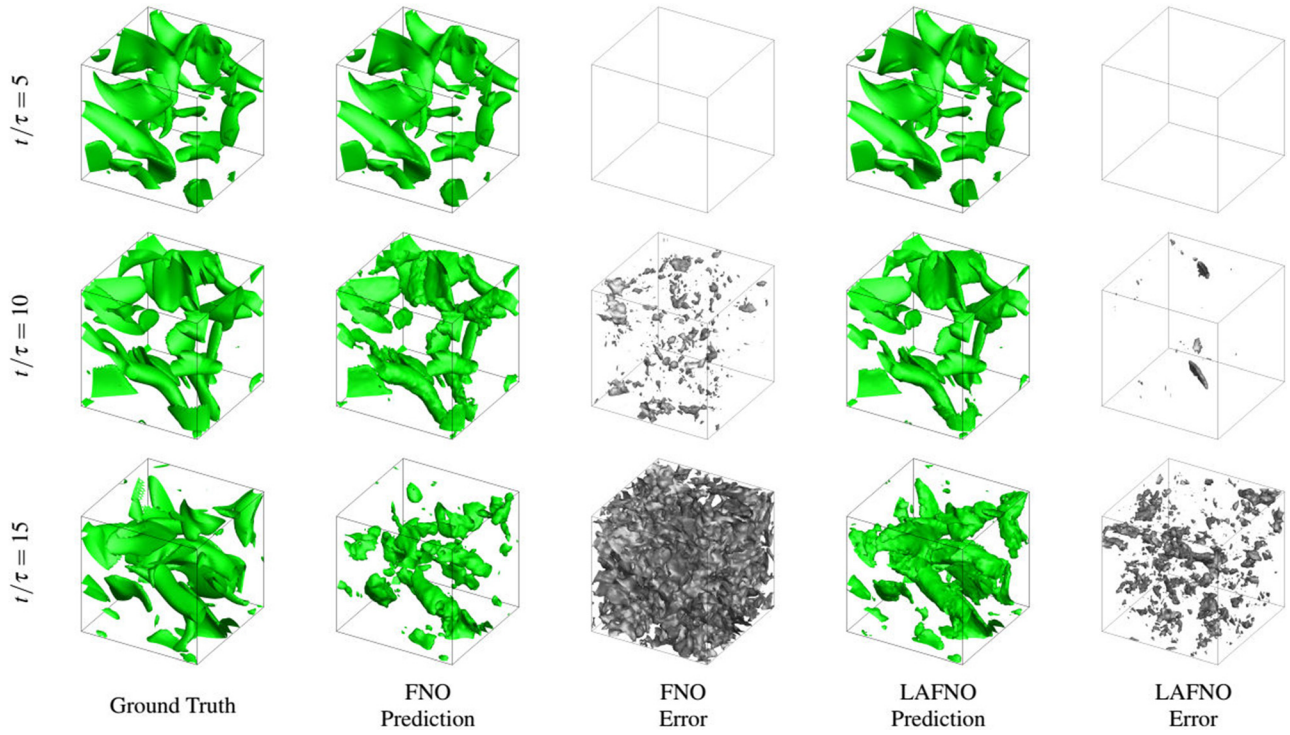

FIG. 8. Isosurfaces of the normalized vorticity magnitude $\Omega/\Omega^{rms} = 1.5$ and relative error $\epsilon = 0.5$ at selected time steps.

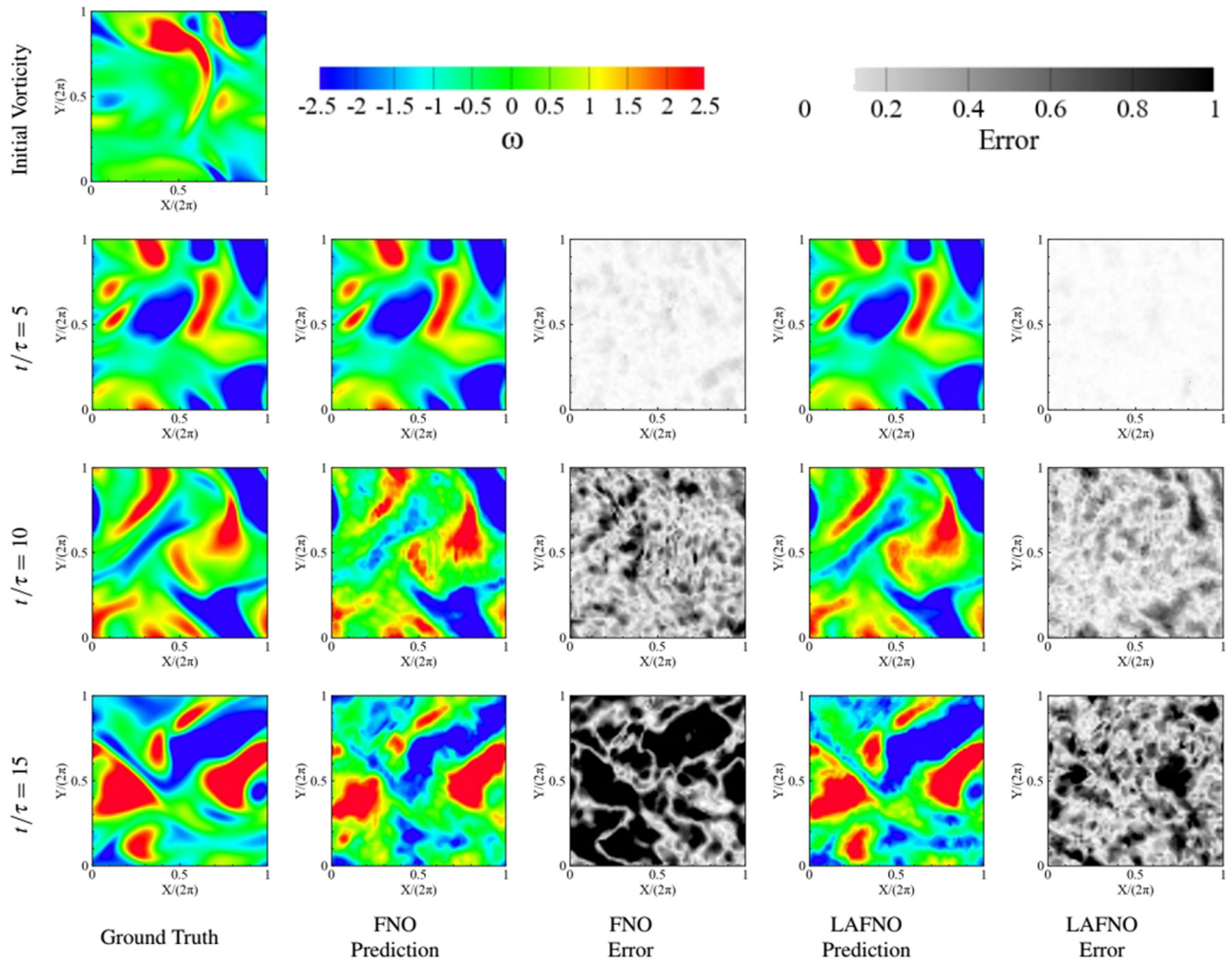

FIG. 9. Vorticity (2D slice) prediction and absolute error at selected time steps.

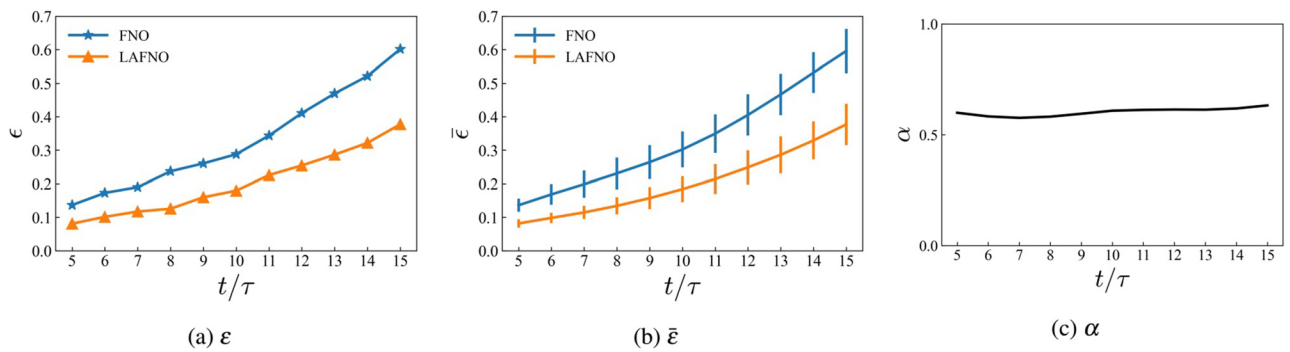

FIG. 10. Relative error comparison at consequent time steps. (a) Spatial-averaged relative error of vorticity on single test sample. (b) Mean and standard deviation of spatial-averaged relative error of vorticity on 100 test samples. (c) Ratio of mean relative error, where $\alpha = \bar{\epsilon}_{LAFNO}/\bar{\epsilon}_{FNO}$.

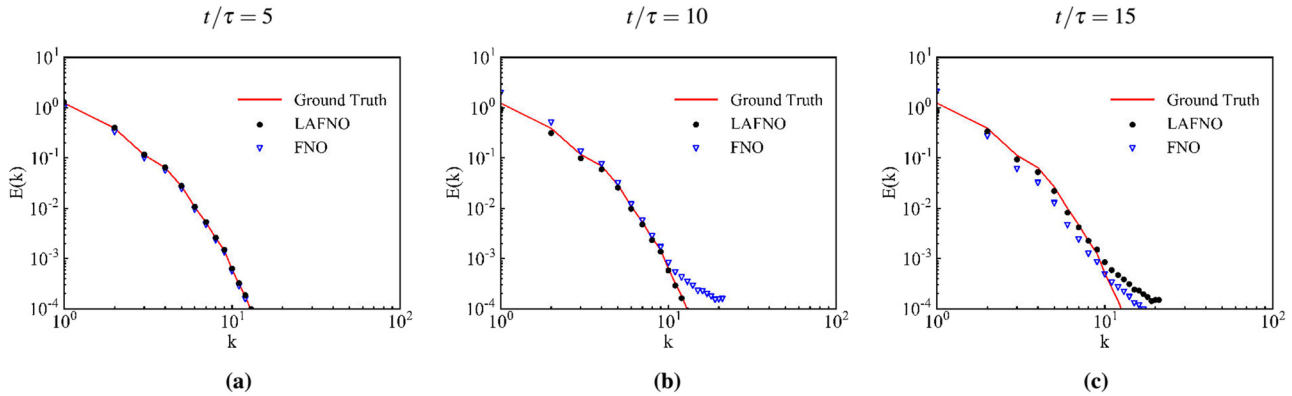

FIG. 11. Averaged velocity spectrum on 100 test samples at (a) $t/\tau = 5$, (b) $t/\tau = 10$, and (c) $t/\tau = 15$.

54% and 32%, respectively. It is also noticed that the LAFNO achieves 40% error reduction compared with FNO, throughout all time steps.

Figure 11 compares the ensemble-averaged velocity spectrum $\overline{E(k)}$ using 100 test samples. The velocity field is calculated from vorticity field by solving the Poission equation $\nabla^2\mathbf{u} = -\nabla \times \boldsymbol{\omega}$. At $t/\tau = 5$, the predicted velocity spectrum of both models can agree well with the ground truth in both the low-wave number region and the high-wave number region. As time advances to $t/\tau = 10$, the FNO predicted spectrum starts to deviate from the ground truth at the high-wave number region. In contrast, the LAFNO can still accurately capture the small-scale flow structures and well reconstruct the velocity spectrum at different flow scales. At $t/\tau = 15$, the predictions of both models deviate from the ground truth at high-wave number region. However, the LAFNO can still make accurate predictions at low-wave number regions whereas the FNO predictions cannot.

Figure 12 shows the probability density functions (PDFs) of the normalized velocity increment $\delta_r u/u^{rms}$ at different time steps, where $\delta_r u = [\mathbf{u}(\mathbf{x}+\mathbf{r}) - \mathbf{u}(\mathbf{x})] \cdot \hat{\mathbf{r}}$ represents the longitudinal increment of the velocity at the separation $\mathbf{r}$. Here, $\hat{\mathbf{r}} = \mathbf{r}/|\mathbf{r}|$, Δ denotes twice the width of the grid. At the beginning of $t/\tau = 5$, the predictions of both models have a good agreement with the ground truth. However, as time advances to $t/\tau = 10$ and $t/\tau = 15$, the predicted PDFs of FNO become more narrower, whereas the predicted PDFs of LAFNO is always consistent with the ground truth.

Figure 13 shows the PDFs of vorticity component ω_z at different time steps. At $t/\tau = 5$, the predicted PDFs of both FNO and LAFNO can agree well with the ground truth. At $t/\tau = 10$, the predicted PDFs of FNO start to deviate from the ground truth. In contrast, the predicted PDFs of LAFNO can still agree well with the ground truth. As time progresses to $t/\tau = 15$, the predicted PDFs of both models deviate from ground truth with LAFNO being significantly closer to the ground truth.

Figure 14 illustrates the PDFs of the normalized characteristic strain-rate, namely, $|S|/|S|^{\text{rms}}_{\text{DNS}}$ at different time steps.[72,75] Here, $|S| = \sqrt{tr(\mathbf{S}^2)}$ and $|S|^{\text{rms}}_{\text{DNS}} = \sqrt{\langle |S|^2_{\text{DNS}} \rangle}$ are, respectively, the characteristic strain rate of the predicted velocity field and the root mean square values of the characteristic strain rate given by the ground truth, where "$tr(\cdot)$" denotes the trace of a matrix and $\mathbf{S} = [\nabla\mathbf{u} + (\nabla\mathbf{u})^T]/2$ stands for the strain-rate tensor of the velocity field.[72,75] At $t/\tau = 5$, the predicted PDFs of LAFNO agree well with the ground truth, while the predicted PDFs of FNO slightly deviates from the ground truth. As time progresses to $t/\tau = 10$ and $t/\tau = 15$, the predicted PDFs of both models deviate from ground truth with LAFNO being significantly closer to the ground truth.

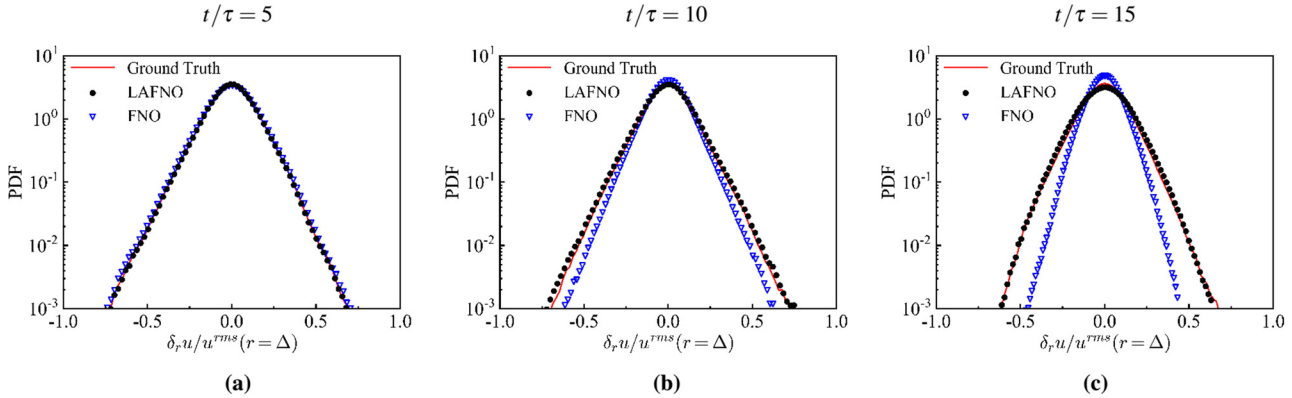

FIG. 12. PDFs of the normalized velocity increments at different time steps on 100 test samples at (a) $t/\tau = 5$, (b) $t/\tau = 10$, and (c) $t/\tau = 15$.

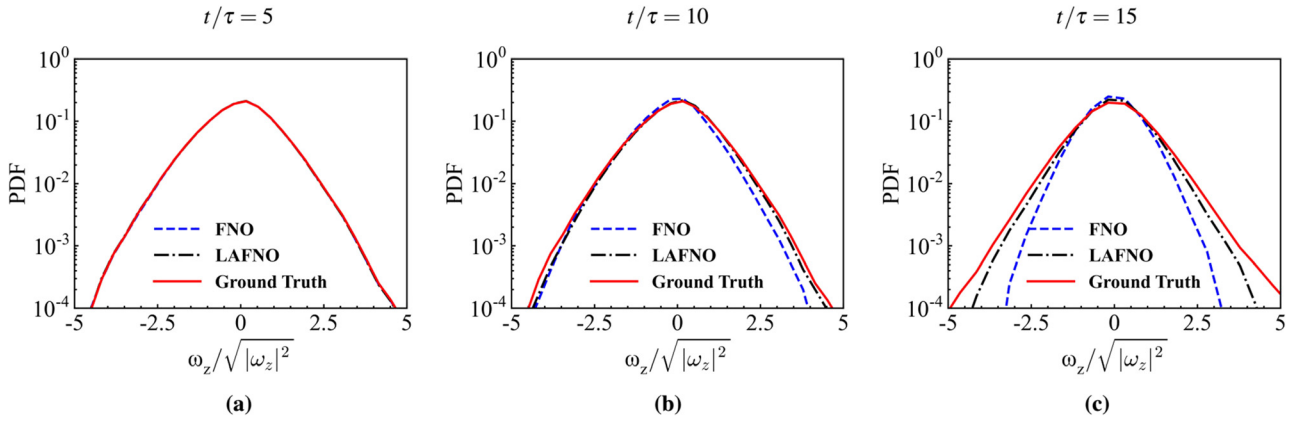

FIG. 13. PDFs of the normalized vorticity component ω_z on 100 test samples at (a) $t/\tau = 5$, (b) $t/\tau = 10$, and (c) $t/\tau = 15$.

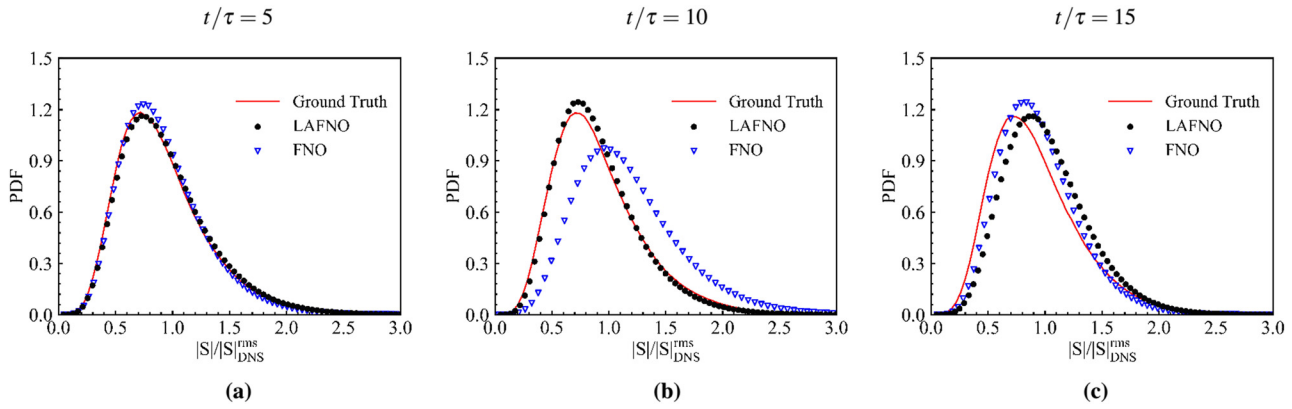

FIG. 14. PDFs of the normalized characteristic strain-rate on 100 test samples at (a) $t/\tau = 5$, (b) $t/\tau = 10$, and (c) $t/\tau = 15$.

A. Generalization on higher Reynolds numbers

Here, we discuss the generalization performance of FNO and LAFNO on higher Reynolds numbers. We train both models on the dataset described in Sec. IV, at Taylor Reynolds number $Re_\lambda = 30$, and evaluate the trained models at $Re_\lambda = 50$ and $Re_\lambda = 70$. Figure 15 shows generalization errors at $t/\tau = 10$ at different Reynolds numbers. It is noted that the prediction errors of both models increase with the increasing of Reynolds numbers. The prediction errors of FNO increase from 0.28 to 0.35 and 0.52, meanwhile, the prediction errors of LAFNO increase from 0.13 to 0.18 and 0.21. The LAFNO performs better than FNO at generalization on higher Reynolds numbers. Moreover, the performance improvement becomes more significant as the Reynolds number gets higher.

Figure 16 compares the ensemble-averaged velocity spectrum $\overline{E(k)}$ at different Reynolds numbers at $t/\tau = 10$ using 100 test samples. At $Re_\lambda = 30$, the predicted velocity spectrum of LAFNO can agree well with the ground truth in both the low- and the high-wave number region, meanwhile, the predicted velocity spectrum of FNO deviates from the ground truth at the high-wave number region. As the Reynolds number increases to $Re_\lambda = 50$, the LAFNO predicted spectrum starts to deviate from the ground truth at the high-wave number region but can still accurately reconstruct the large-scale flow structures at low-wave number region. In contrast, the FNO predicted spectrum deviates from the ground truth at both the low- and the high-wave number region. At $Re_\lambda = 70$, the predictions of both

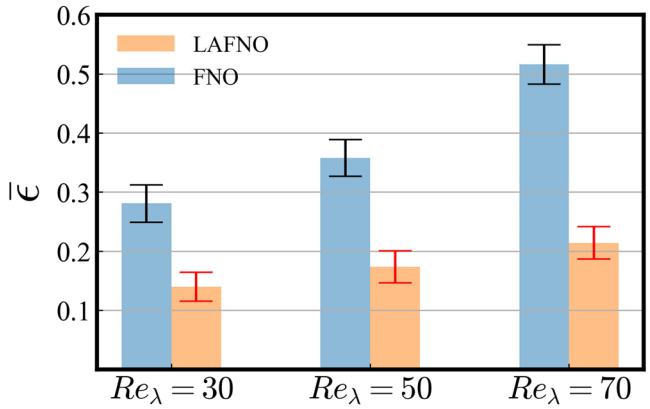

FIG. 15. Mean and standard deviation of spatial-averaged relative error of vorticity on 100 test samples at $t/\tau = 10$ for three different Taylor Reynolds numbers.

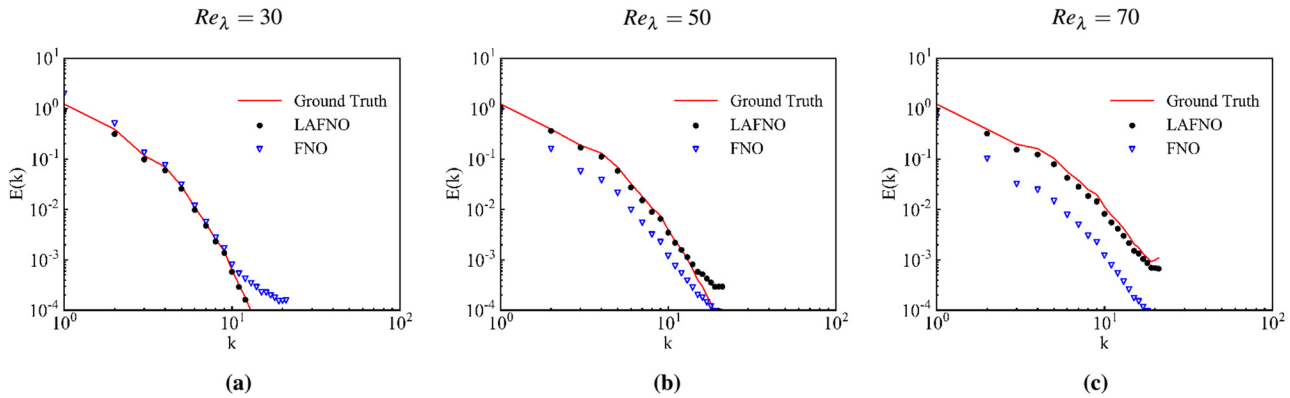

FIG. 16. Averaged velocity spectrum on 100 test samples at $t/\tau = 10$ for Taylor Reynolds numbers of (a) $Re_\lambda = 30$, (b) $Re_\lambda = 50$, and (c) $Re_\lambda = 70$.

models deviate from the ground truth, with LAFNO being significantly closer to the ground truth.

B. Performance on free shear turbulence

In addition to 3D homogeneous isotropic turbulence, we also benchmark the performance of FNO and LAFNO on a more complex turbulence simulation task: the 3D free shear turbulence. The 3D free shear turbulence is governed by the same Navier–Stokes equations displayed in Eqs. (10) and (11) without the forcing term.[76,77] The simulations of free shear turbulence are performed with lengths $L_1 \times L_2 \times L_3 = 8\pi \times 8\pi \times 4\pi$. x_1, x_2, and x_3, respectively, denotes the streamwise, normal, and spanwise directions, using the uniform grids with $N_1 \times N_2 \times N_3 = 64 \times 64 \times 32$.[76] Similar to the 3D homogeneous isotropic turbulence, the periodic boundary conditions in all three directions are adopted and an explicit two-step Adam–Bashforth scheme is selected as the time marching scheme. The Reynolds number is $Re = 2000$. The initial conditions of 3D free shear turbulence are given by the following equation:[76,77]

$$u_1 = \frac{\Delta U}{2}\left[\tanh\left(\frac{x_2}{2\delta_\theta^0}\right) - \tanh\left(\frac{x_2 + L_2/2}{2\delta_\theta^0}\right) - \tanh\left(\frac{x_2 - L_2/2}{2\delta_\theta^0}\right)\right] + \lambda_1,$$
$$u_2 = \lambda_2, \quad u_3 = \lambda_3, \tag{20}$$

where ΔU is the free-stream velocity difference across the shear layer. Here, the magnitudes of perturbation λ_1, λ_2, and λ_3 satisfy the Gaussian random distribution where $\lambda_1, \lambda_2, \lambda_3 \sim \mathcal{N}(0, 10^{-2})$, and $\delta_\theta^0 = 0.08$ is the initial momentum thickness.[76,77]

We generate 3000 pairs of input-output data with the numerical solver, where each sample contains 15 steps of solutions of a random initialized condition. We use 2400 samples for training and 600 samples for testing. After training, we evaluate both models on the test dataset and compare their performance at three selected time steps $t = 5$, 10, and 15, corresponding to the dimensionless time $t/\tau' = 125$, 250, and 375, respectively, where $\tau' \equiv \delta_\theta^0/\Delta U = 0.04$.[72,76,77]

Figure 17 shows the spatial-averaged relative errors of the two models with respect to consequent time steps. Both models can make accurate predictions in the beginning ($t/\tau' = 125$) with about 10% error. However, since the predictions at each step is recurrently treated as ground truth and reused as the inputs with the advance of time, the prediction errors of both models accumulate iteratively. As time progresses to $t/\tau' = 375$, the mean errors of FNO and LAFNO increase to 52% and 30%, respectively. The LAFNO performs better than FNO throughout all the time steps.

Figure 18 shows a 2*D* vorticity slice in the middle of *Y* axis from a test sample. At $t/\tau' = 125$, both models can accurately reconstruct the instantaneous spatial structures of turbulence. As time progresses to $t/\tau' = 250$, the difference can be visibly noticed: the LAFNO can make better reconstructions on the small-scale structures than FNO. At $t/\tau' = 375$, the LAFNO can still make relatively accurate reconstructions on the large-scale instantaneous structures, whereas the FNO cannot. The errors of LAFNO are visibly smaller than FNO in terms of both the magnitude and region.

Figure 19 shows the PDFs of vorticity component ω_z at different time steps on 100 test samples. At $t/\tau' = 125$, the predicted PDFs of LAFNO can agree well with the ground truth, while the predicted PDFs of FNO deviate from the ground truth. As time progresses to $t/\tau' = 250$ and $t/\tau' = 375$, the predicted PDFs of FNO deviate

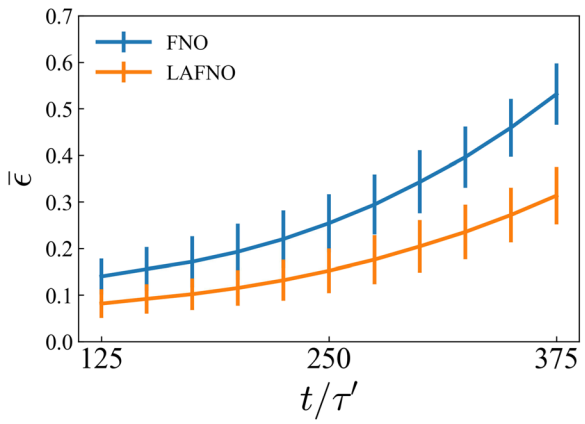

FIG. 17. Mean and standard deviation of spatial-averaged relative error of vorticity in free shear turbulence on 100 test samples at different time steps.

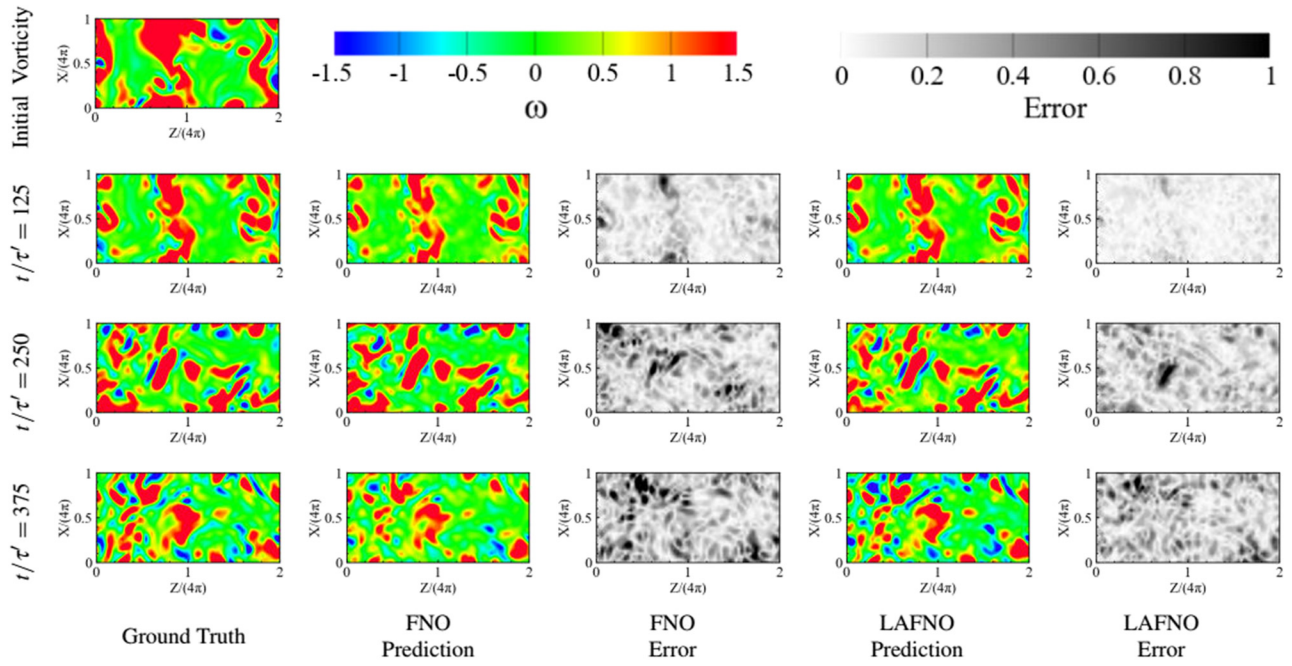

FIG. 18. Vorticity (2D slice) prediction and absolute error in free shear turbulence at selected time steps.

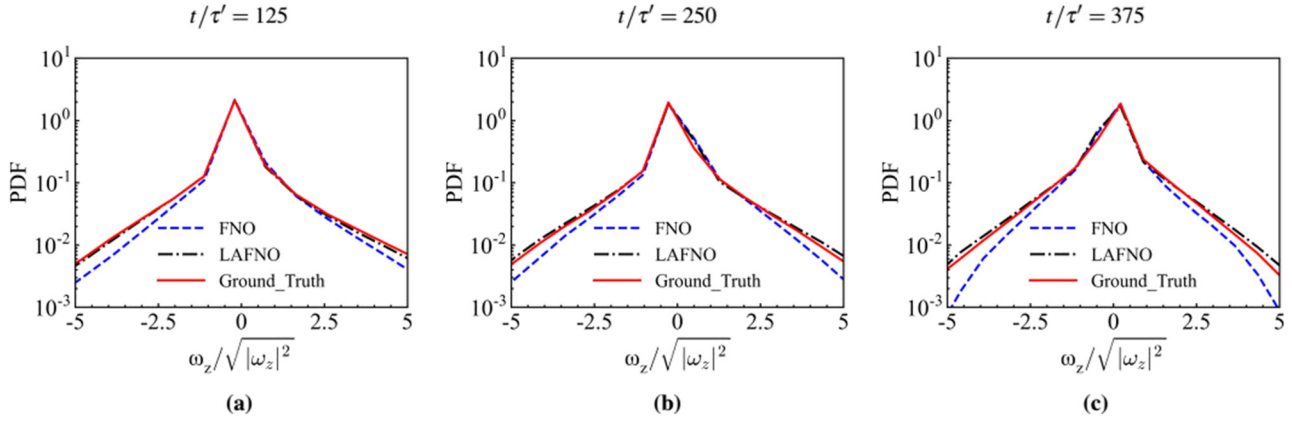

FIG. 19. PDFs of the normalized vorticity component ω_z in free shear turbulence on 100 test samples at (a) $t/\tau' = 125$, (b) $t/\tau' = 250$, and (c) $t/\tau' = 375$.

remarkably from the ground truth, with LAFNO being significantly closer to the ground truth.

C. Computational efficiency

Table IV compares computational cost of ten prediction steps on a $64 \times 64 \times 64$ grid using three different approaches. We implement the numerical experiments on the Pytorch and MindSpore open-source deep learning frameworks. The neural network models (FNO, LAFNO) are trained and tested on Nvidia Tesla A100 GPU, where the CPU type is Intel Xeon(R) Platinum 8350 C @ 2.60GHz.

TABLE IV. Computational efficiency comparison of three different numerical approaches.

Method	Parameters	Training time (h)	Inference time (s)
Numerical solver	N/A	N/A	12.53
LAFNO	70, 570, 874	168 h	0.576
FNO	69, 605, and 997	170 h	0.538

The traditional numerical solver is ran on a computing cluster, where the CPU type is 2-sockets Intel Xeon Gold 6148 with 20 cores each @2.40GHz for a total of 40 cores per node. The time consumption of training both models is comparable. Training FNO takes 170 h, for 500 epochs, where each epoch takes 0.34 h. On the other hand, training LAFNO takes 168 h, for 300 epochs, with 0.56 h per epoch. The LAFNO needs less training epochs than FNO, because LAFNO converges faster, as shown in Fig. 7. Once trained, both models can make efficient predictions to any random initial conditions. During inference, both neural network models provides 20 folds speedup compared with the DNS approach with the traditional numerical solver. Moreover, the LAFNO achieves 40% error reduction at the same level of memory consuming and computational expense as compared to FNO.

VII. DISCUSSION AND FUTURE WORK

Problems in scientific computations and engineering applications often involve solving 3D nonlinear PDEs. However, most existing data-driven approaches have only focused on solving one dimensional (1D) and 2D PDEs, and the 3D problems are rarely discussed and explored. One of the most important reason is that modeling 3D nonlinear PDEs with deep neural networks can be computationally expensive. The size and dimension of simulation data increases dramatically from 2D to 3D. Moreover, modeling such high-dimensional data requires huge number of parameters with hundreds of layers not being uncommon.[51] Training and deploying such neural networks can be inefficient when compared with traditional numerical approaches. The FNO has been shown to be one of the most efficient surrogate models in solving PDEs,[25,41,43] thus, can be very potential in dealing with 3D nonlinear problems.

Recently, the attention mechanism has been shown to be very promising in boosting the performance of neural networks on solving PDEs.[58–61] Peng *et al.* coupled the self-attention mechanism with FNO, to enhance the FNO prediction ability on 2*D* turbulence simulation.[40] The attention mechanism provides 40% prediction error reduction compared with the original FNO.[40]

However, these works are limited to 2D turbulence simulations, and extending the attention mechanism to 3D turbulence simulation is a non-trivial task. The challenge comes from the computational expense of the self-attention matrix: the standard self-attention mechanism uses $O(n^2)$ time and space with respect to input dimension n.[55] For a typical 3 D flow field of grid size $64 \times 64 \times 64$, computing the attention matrix requires 2034 GB memory for 32-bit floating point data type. Such prohibitively computational cost has become the main bottleneck for the attention mechanism to be applied on 3D turbulence simulation.

In this work, we explore the possibility of coupling attention with FNO for 3D turbulence simulation. With the linear attention approximation, the memory consumption can be reduced to 35.82 GB, thus allowing the attention mechanism to be coupled with FNO. Our results show that the linear attention is very effective in boosting the performance of FNO on predicting simple 3D turbulent flows, including 3D homogeneous isotropic turbulence and free shear turbulence. Moreover, the attention mechanism can significantly improve the generalization ability of FNO at higher Reynolds numbers. The proposed linear attention coupled FNO provides an important reference for accelerating 3D complex turbulence simulations.

One limitation of the proposed linear attention is that the problem of error accumulation over time is improved but not fundamentally resolved. In order to solve this problem, physical constraints can be incorporated to ensure that the predictions are subject to the governing equations and conservation laws.[21,27,45–48]

Another limitation is that the FNO architecture has only been validated on simple flows including homogeneous isotropic turbulence and free shear turbulence, while the actual engineering flows are usually more complex. Recently, some advanced variants of the FNO architecture have been developed to model complex flows. Improved models including factorized Fourier neural operators (FFNO),[78] physics-informed neural operator (PINO),[45] adaptive Fourier neural operators (AFNO),[38] and U-shaped neural operators (UNO)[79] have been proposed to simulate complex 2D flows. Li *et al.* proposed the Geo-FNO,[43] to solve PDEs on arbitrary geometries. The proposed Geo-FNO learns to deform the irregular input domain, into a latent space with a uniform grid, and performs fast Fourier transform on the uniform grid. Such flexibility of handling arbitrary geometries is crucial for solving engineering flows, which usually have more complex geometries with irregular boundaries. However, these advanced FNO variants are still mainly focused on 2D problem, whereas the engineering flows are usually 3D. These FNO variants, including the Geo-FNO, could be extended and coupled with the linear attention and physical constraints to effectively simulate the 3D engineering complex flows in the future work.

VIII. CONCLUSION

In this work, we apply the linear attention mechanism to improve neural network models on fast simulation of three-dimensional turbulence. The linear attention approximation reduces the overall self-attention complexity from $O(n^2)$ to $O(n)$ in both time and space, allowing the attention mechanism to be coupled with neural networks and trained on GPUs for 3D turbulence problem. The linear attention coupled Fourier neural operator (LAFNO) is developed and tested for the simulations of 3D turbulence, including homogeneous isotropic turbulence and free shear turbulence.

Numerical experiments show that (1) the LAFNO can accurately reconstruct a variety of statistics and instantaneous spatial structures of 3D turbulence. (2) The linear attention can reduce 40% of the prediction error throughout all the time steps. (3) The linear attention coupled FNO generalizes better at higher Reynolds numbers than the original FNO. (4) The linear attention coupled FNO model achieves the same level of computational efficiency as compared with the original FNO model. In addition to 3D turbulence simulations, the linear attention can be helpful for the development of advanced neural network models of other 3D nonlinear problems with high-dimensional data.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC Grant Nos. 91952104, 92052301, 12172161, and 91752201), the National Numerical Wind Tunnel Project (No. NNW2019ZT1-A04), the Shenzhen Science and Technology Program (Grant No. KQTD20180411143441009), the Key Special Project for Introduced Talents Team of Southern Marine Science and Engineering Guangdong Laboratory (Guangzhou) (Grant No. GML2019ZD0103), the CAAI-Huawei

MindSpore Open Fund, and the Department of Science and Technology of Guangdong Province (No. 2020B1212030001). This work was also supported by Center for Computational Science and Engineering of Southern University of Science and Technology.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Wenhui Peng: Conceptualization (equal); Investigation (equal); Methodology (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Zelong Yuan:** Conceptualization (equal); Formal analysis (equal); Visualization (equal); Writing – review & editing (equal). **Zhijie Li:** Validation (equal); Visualization (equal). **Jianchun Wang:** Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Project administration (equal); 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

[1]S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477 (2020).
[2]K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence modeling in the age of data," Annu. Rev. Fluid Mech. **51**, 357 (2019).
[3]R. Maulik, O. San, A. Rasheed, and P. Vedula, "Subgrid modelling for two-dimensional turbulence using neural networks," J. Fluid Mech. **858**, 122 (2019).
[4]J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," J. Fluid Mech. **807**, 155 (2016).
[5]A. Beck, D. Flad, and C.-D. Munz, "Deep neural networks for data-driven LES closure models," J. Comput. Phys. **398**, 108910 (2019).
[6]Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436 (2015).
[7]Y. Guan, A. Chattopadhyay, A. Subel, and P. Hassanzadeh, "Stable *a posteriori* LES of 2D turbulence using convolutional neural networks: Backscattering analysis and generalization to higher *Re* via transfer learning," J. Comput. Phys. **458**, 111090 (2022).
[8]X. Yang, S. Zafar, J.-X. Wang, and H. Xiao, "Predictive large-eddy-simulation wall modeling via physics-informed neural networks," Phys. Rev. Fluids **4**, 034602 (2019).
[9]B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," Nat. Commun. **9**(1), 4950 (2018).
[10]J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," J. Comput. Phys. **375**, 1339 (2018).
[11]H. S. Tang, L. Li, M. Grossberg, Y. Liu, Y. M. Jia, S. S. Li, and W. B. Dong, "An exploratory study on machine learning to couple numerical solutions of partial differential equations," Commun. Nonlinear Sci. Numer. Simul. **97**, 105729 (2021).
[12]Y. Sun, L. Zhang, and H. Schaeffer, "NeuPDE: Neural network based ordinary and partial differential equations for modeling time-dependent data," in *Mathematical and Scientific Machine Learning PMLR* (PMLR, 2020), pp. 352–372.
[13]N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Learning maps between function spaces," arXiv:2108.08481 (2021).
[14]X. Meng, L. Yang, Z. Mao, J. del Águila Ferrandis, and G. E. Karniadakis, "Learning functional priors and posteriors from data and physics," J. Comput. Phys. **457**, 111073 (2022).
[15]K. Linka, A. Schafer, X. Meng, Z. Zou, G. E. Karniadakis, and E. Kuhl, "Bayesian physics-informed neural networks for real-world nonlinear dynamical systems," arXiv:2205.08304 (2022).
[16]S. Goswami, K. Kontolati, M. D. Shields, and G. E. Karniadakis, "Deep transfer learning for partial differential equations under conditional shift with DeepONet," arXiv:2204.09810 (2022).
[17]A. A. Howard, M. Perego, G. E. Karniadakis, and P. Stinis, "Multifidelity deep operator networks," arXiv:2204.09157 (2022).
[18]Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Graph kernel network for partial differential equations," arXiv:2003.03485 (2020).
[19]H. Xu, W. Zhang, and Y. Wang, "Explore missing flow dynamics by physics-informed deep learning: The parameterized governing systems," Phys. Fluids **33**, 095116 (2021).
[20]R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (ACM SIGKDD, 2020), pp. 1457–1466.
[21]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686 (2019).
[22]S. Pan and K. Duraisamy, "Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability," SIAM J. Appl. Dyn. Syst. **19**, 480 (2020).
[23]K. Wu and D. Xiu, "Data-driven deep learning of partial differential equations in modal space," J. Comput. Phys. **408**, 109307 (2020).
[24]H. Xu, D. Zhang, and J. Zeng, "Deep-learning of parametric partial differential equations from sparse and noisy data," Phys. Fluids **33**, 037132 (2021).
[25]Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," arXiv:2010.08895 (2020).
[26]Y. Fan, C. O. Bohorquez, and L. Ying, "BCR-Net: A neural network based on the nonstandard wavelet form," J. Comput. Phys. **384**, 1–15 (2019).
[27]K. Kashinath, P. Marcus, *et al.*, "Enforcing physical constraints in CNNs through differentiable PDE layer," in ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020).
[28]J. Chen, J. Viquerat, and E. Hachem, "U-net architectures for fast prediction of incompressible laminar flows," arXiv:1910.13532 (2019).
[29]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR, 2016), pp. 770–778.
[30]V. Sekar, Q. Jiang, C. Shu, and B. C. Khoo, "Fast flow field prediction over airfoils using deep learning approach," Phys. Fluids **31**, 057103 (2019).
[31]C. Cheng and G.-T. Zhang, "Deep learning method based on physics informed neural network with Resnet block for solving fluid flow problems," Water **13**, 423 (2021).
[32]M. Z. Yousif, L. Yu, and H.-C. Lim, "High-fidelity reconstruction of turbulent flow from spatially limited data using enhanced super-resolution generative adversarial network," Phys. Fluids **33**, 125119 (2021).
[33]K. Hasegawa, K. Fukami, T. Murata, and K. Fukagata, "CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers," Fluid Dyn. Res. **52**, 065501 (2020).
[34]K. Hasegawa, K. Fukami, T. Murata, and K. Fukagata, "Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes," Theor. Comput. Fluid Dyn. **34**, 367 (2020).
[35]J. Chen, E. Hachem, and J. Viquerat, "Graph neural networks for laminar flow prediction around random two-dimensional shapes," Phys. Fluids **33**, 123607 (2021).
[36]A. Patil, J. Viquerat, and E. Hachem, "Autoregressive transformers for data-driven spatio-temporal learning of turbulent flows," arXiv:2209.08052 (2022).
[37]G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson, "U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow," Adv. Water Resour. **163**, 104180 (2022).

[38]J. Guibas, M. Mardani, Z. Li, A. Tao, A. Anandkumar, and B. Catanzaro, "Adaptive Fourier neural operators: Efficient token mixers for transformers," arXiv:2111.13587 (2021).
[39]J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, *et al.*, "FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators," arXiv:2202.11214 (2022).
[40]W. Peng, Z. Yuan, and J. Wang, "Attention-enhanced neural network models for turbulence simulation," Phys. Fluids **34**, 025111 (2022).
[41]Z. Li, K. Meidani, and A. B. Farimani, "Transformer for partial differential equations' operator learning," arXiv:2205.13671 (2022).
[42]X. Ye, H. Li, P. Jiang, T. Wang, and G. Qin, "Learning transient partial differential equations with local neural operators," arXiv:2203.08145 (2022).
[43]Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar, "Fourier neural operator with learned deformations for PDEs on general geometries," arXiv:2207.05209 (2022).
[44]A. F. Psaros, K. Kawaguchi, and G. E. Karniadakis, "Meta-learning PINN loss functions," J. Comput. Phys. **458**, 111121 (2022).
[45]Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, "Physics-informed neural operator for learning partial differential equations," arXiv:2111.03794 (2021).
[46]S. Goswami, A. Bora, Y. Yu, and G. E. Karniadakis, "Physics-informed deep neural operators networks," arXiv:2207.05748 (2022).
[47]X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations," J. Comput. Phys. **426**, 109951 (2021).
[48]A. Kashefi and T. Mukerji, "Physics-informed PointNet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries," arXiv:2202.05476 (2022).
[49]M. Momenifar, E. Diao, V. Tarokh, and A. D. Bragg, "Dimension reduced turbulent flow data from deep vector quantisers," J. Turbul. **23**, 232 (2022).
[50]A. Glaws, R. King, and M. Sprague, "Deep learning for *in situ* data compression of large turbulent flow simulations," Phys. Rev. Fluids **5**, 114602 (2020).
[51]F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Local binary convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR, 2017), pp. 19–28.
[52]A. T. Mohan, D. Tretiak, M. Chertkov, and D. Livescu, "Spatio-temporal deep learning models of 3D turbulence with physics informed diagnostics," J. Turbul. **21**, 484 (2020).
[53]X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems* (NIPS, 2015), p. 28.
[54]T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, and K. Fukagata, "Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow," Phys. Fluids **33**, 025116 (2021).
[55]A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (NIPS, 2017), pp. 5998–6008.
[56]N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International Conference on Machine Learning PMLR* (PMLR, 2018), pp. 4055–4064.
[57]X. Liu and M. Milanova, "Visual attention deep learning: A review," Int. Rob. Autom. J. **4**, 154 (2018).
[58]P. Wu, S. Gong, K. Pan, F. Qiu, W. Feng, and C. Pain, "Reduced order model using convolutional auto-encoder with self-attention," Phys. Fluids **33**, 077107 (2021).
[59]I. K. Deo and R. Jaiman, "Learning wave propagation with attention-based convolutional recurrent autoencoder net," arXiv:2201.06628 (2022).
[60]Q. Liu, W. Zhu, F. Ma, X. Jia, Y. Gao, and J. Wen, "Graph attention network-based fluid simulation model," AIP Adv. **12**, 095114 (2022).
[61]G. Kissas, J. H. Seidman, L. F. Guilhoto, V. M. Preciado, G. J. Pappas, and P. Perdikaris, "Learning operators with coupled attention," J. Mach. Learn. Res. **23**, 1–63 (2022).
[62]S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," arXiv:2006.04768 (2020).
[63]B. Beauzamy, *Introduction to Banach Spaces and Their Geometry* (Elsevier, 2011).
[64]V. N. Vapnik, "An overview of statistical learning theory," IEEE Trans. Neural Networks **10**, 988 (1999).
[65]G. J. Kowalski, *Information Retrieval Systems: Theory and Implementation* (Springer, 2007), Vol. 1.
[66]A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, "Training with quantization noise for extreme model compression," arXiv:2004.07320 (2020).
[67]R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," arXiv:1904.10509 (2019).
[68]N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," arXiv:2001.04451 (2020).
[69]W. B. Johnson, "Extensions of Lipschitz mappings into a Hilbert space," Contemp. Math. **26**, 189 (1984).
[70]F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR, 2017) pp. 3156–3164.
[71]S. B. Pope and S. B. Pope, *Turbulent Flows* (Cambridge University Press, 2000).
[72]Z. Yuan, Y. Wang, C. Xie, and J. Wang, "Dynamic iterative approximate deconvolution models for large-eddy simulation of turbulence," Phys. Fluids **33**, 085125 (2021).
[73]J. Wang, M. Wan, S. Chen, C. Xie, Q. Zheng, L.-P. Wang, and S. Chen, "Effect of flow topology on the kinetic energy flux in compressible isotropic turbulence," J. Fluid Mech. **883**, A11 (2020).
[74]Z. Yuan, C. Xie, and J. Wang, "Deconvolutional artificial neural network models for large eddy simulation of turbulence," Phys. Fluids **32**(11), 115106 (2020).
[75]Z. Yuan, Y. Wang, C. Xie, and J. Wang, "Dynamic nonlinear algebraic models with scale-similarity dynamic procedure for large-eddy simulation of turbulence," Adv. Aerodyn. **4**, 16 (2022).
[76]X. Wang, J. Wang, and S. Chen, "Compressibility effects on statistics and coherent structures of compressible turbulent mixing layers," J. Fluid Mech. **947**, A38 (2022).
[77]Y. Wang, Z. Yuan, X. Wang, and J. Wang, "Constant-coefficient spatial gradient models for the sub-grid scale closure in large-eddy simulation of turbulence," Phys. Fluids **34**, 095108 (2022).
[78]A. Tran, A. Mathews, L. Xie, and C. S. Ong, "Factorized Fourier neural operators," arXiv:2111.13802 (2021).
[79]M. A. Rahman, Z. E. Ross, and K. Azizzadenesheli, "U-No: U-shaped neural operators," arXiv:2204.11127 (2022).