



Titre: Applications of Deep Learning in Fluid Dynamics
Title:

Auteur: Wenhui Peng
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Peng, W. (2023). Applications of Deep Learning in Fluid Dynamics [Thèse de doctorat, Polytechnique Montréal]. PolyPublie.
Citation: <https://publications.polymtl.ca/54933/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/54933/>
PolyPublie URL:

Directeurs de recherche: Michel Desmarais, & Éric Laurendeau
Advisors:

Programme: Génie informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Applications of deep learning in fluid dynamics

WENHUI PENG

Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*

Génie informatique

Juillet 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Cette thèse intitulée :

Applications of deep learning in fluid dynamics

présentée par **Wenhui PENG**

en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

François GUIBAULT, président

Michel DESMARAIS, membre et directeur de recherche

Éric LAURENDEAU, membre et codirecteur de recherche

David VIDAL, membre

Quentin DOLLON, membre externe

DEDICATION

To my beloved family, and friends at the lab

...

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor Professor Michel Desmarais. Michel has offered generous support during the four years of my PhD. He is more than just a supervisor for academics, but also a wise friend for life.

Then I would like to thank my co-supervisor Professor Eric Laurendeau, he provided help with his expertise at the end of my PhD, especially in the field of aerodynamics.

I would also like to thank my family and my friends in the lab, I could not have completed my study without the support from my family and friends, who never stopped encouraging me to strive for success.

RÉSUMÉ

Avec l’augmentation de la puissance de calcul ces dernières années, des méthodes basées sur les données et des approches d’apprentissage automatique ont été appliquées dans de nombreux domaines d’ingénierie pratiques pour aider à la conception et à l’optimisation, y compris la dynamique des fluides et l’aérodynamique. Les réseaux de neurones (NN) se sont révélés très efficaces pour approximer les fonctions non linéaires. La nature hautement non linéaire des systèmes dynamiques des fluides fournit un scénario idéal pour les applications interdisciplinaires avec les NN. Au cours des dernières années, les approches d’apprentissage en profondeur ont montré un potentiel important pour compenser les lacunes de la modélisation traditionnelle de la dynamique des fluides computationnelle (CFD).

Dans la première partie, nous introduisons une architecture de réseau de neurones de bout en bout, l’Element Spatial Convolutional Neural Network (ESCNN), pour la prédiction du coefficient de portance du profil aérodynamique. L’ESCNN surpasse les NN de pointe existants en termes de précision de prédiction, avec deux ordres de grandeur moins de paramètres. Nous montrons en outre comment ESCNN réussit à faire des prédictions précises. Nous avons étudié une couche du modèle ESCNN au cours du processus de formation, et nous avons découvert que le NN apprend des modèles qui émergent de l’aérodynamique, et ces modèles sont clairement reflétés dans l’une des couches du NN. Nous montrons que l’ESCNN apprend les lois physiques et les équations de l’aérodynamique, et fait des inférences basées sur les connaissances physiques apprises.

Dans la deuxième partie, nous couplons le mécanisme d’auto-attention avec le modèle de réseau neuronal de pointe, l’opérateur neuronal de Fourier, pour améliorer la précision de la prédiction sur la tâche de simulation de turbulence bidimensionnelle. Des expériences numériques montrent que l’opérateur neuronal de Fourier renforcé par l’attention surpasse les lignes de base de pointe existantes et peut reconstruire avec précision une variété de statistiques et de structures spatiales instantanées de turbulence à des nombres de Reynolds élevés. De plus, le mécanisme d’attention fournit une réduction d’erreur de 40% avec une augmentation de 1% des paramètres, au même niveau de coût de calcul.

Dans la troisième partie, nous étendons le mécanisme d’auto-attention à la tâche de simulation de turbulence tridimensionnelle. Le mécanisme d’attention s’est révélé être une approche prometteuse pour améliorer les performances des réseaux de neurones sur la simulation de la turbulence. Cependant, le mécanisme standard d’auto-attention utilise le temps et l’espace $O(n^2)$ par rapport à la dimension d’entrée n , et une telle complexité quadratique est devenue

le principal goulot d'étranglement pour l'attention à appliquer sur la simulation de turbulence 3D. Nous résolvons ce problème avec le concept de réseau d'attention linéaire. L'attention linéaire se rapproche de l'attention standard en ajoutant deux projections linéaires, réduisant la complexité globale de l'attention de soi de $O(n^2)$ à $O(n)$ à la fois dans le temps et dans l'espace, permettant ainsi de coupler le mécanisme de l'attention avec des réseaux de neurones et formés sur des GPU. Des simulations numériques montrent que le mécanisme d'attention linéaire permet une réduction d'erreur de 40% au même niveau de coût de calcul.

ABSTRACT

With the rise of computational power in recent years, data driven methods and machine learning approaches have been applied over numerous practical engineering fields to assist the design and optimization including fluid dynamics and aerodynamics. Neural Networks (NNs) have shown themselves to be very successful at approximating nonlinear functions. The highly-nonlinear nature of fluid dynamic systems provides an ideal scenario for interdisciplinary applications with NNs. Over the past few years, deep learning approaches have shown significant potential to compensate the shortcomings of traditional Computational Fluid Dynamics (CFD) modeling.

In the first part, we introduce an end-to-end neural network architecture, the Element Spatial Convolutional Neural Network (ESCNN), for the prediction of airfoil lift coefficient. The ESCNN outperforms existing state-of-the-art NNs in terms of prediction accuracy, with two orders of magnitude less parameters. We further show how ESCNN succeeds in making accurate predictions. We investigated a layer of the ESCNN model during the training process, and we discovered that the NN learns patterns that emerge from aerodynamics, and such patterns are clearly reflected within one of the layers of the NN. We show that the ESCNN learns the physical laws and equations of aerodynamics, and makes inferences based on the learned physical knowledge.

In the second part, we couple the self-attention mechanism with the state-of-the-art neural network model, the Fourier neural operator, to improve the prediction accuracy on two dimensional turbulence simulation task. Numerical experiments show that the attention-enhanced Fourier neural operator outperforms existing state-of-the-art baselines, and can accurately reconstruct a variety of statistics and instantaneous spatial structures of turbulence at high Reynolds numbers. Furthermore, the attention mechanism provides 40% error reduction with 1% increase in parameters, at the same level of computational cost.

In the third part, we extend the self-attention mechanism to three-dimensional turbulence simulation task. 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. We resolve this issue with the concept of 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,

thus allowing the attention mechanism to be coupled with neural networks and trained on GPUs. Numerical simulations show that the linear attention mechanism provides 40% error reduction at the same level of computational cost.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vii
TABLE OF CONTENTS	ix
LIST OF TABLES	xii
LIST OF FIGURES	xiii
LIST OF SYMBOLS AND ACRONYMS	xvi
CHAPTER 1 INTRODUCTION	1
1.1 Two-dimensional inviscid flow	2
1.2 Two-dimensional viscous flow	3
1.3 Three-dimensional viscous flow	5
1.4 Data description and general methodology	6
CHAPTER 2 LITERATURE REVIEW	9
2.1 Neural networks for fluid dynamics	9
2.2 Convolution neural network	10
2.3 Predicting the airfoil lift coefficient in the two-dimensional inviscid flow	11
2.4 Two-dimensional turbulent flow prediction with NNs	13
2.4.1 The Fourier neural operator	14
2.4.2 Self-attention mechanism	15
2.4.3 Self-attention enhanced FNO	17
2.5 3D turbulence simulation with NNs	19
2.5.1 Prediction accuracy issue	19
2.5.2 Self-attention quadratic complexity	20
2.5.3 Linear self-attention coupled FNO for 3D turbulence	22
CHAPTER 3 OBJECTIVES AND CONTRIBUTIONS	25

3.1	Structure of the dissertation	25
CHAPTER 4 ARTICLE 1: LEARNING AERODYNAMICS WITH NEURAL NETWORK		
	WORK	28
4.1	Introduction	28
4.2	Predicting lift coefficient with neural networks	29
4.3	Learning the Kutta condition	31
4.4	Learning the vortex strength distribution pattern	33
4.5	Learning with ReLU activation	36
4.6	Learning the lift coefficient equation	37
4.7	Conclusions	39
CHAPTER 5 ARTICLE 2: ATTENTION-ENHANCED NEURAL NETWORK MODELS FOR TURBULENCE SIMULATION		
	ELS	41
5.1	Introduction	41
5.2	The Fourier neural operator	43
5.3	Attention-enhanced neural network	45
5.4	Performance Benchmark	46
5.4.1	Dataset description	48
5.4.2	Performance benchmark at different time steps	49
5.4.3	Performance benchmark at different Reynolds numbers	51
5.4.4	Zero-shot super-resolution benchmark	57
5.4.5	Performance benchmark on Taylor-Green turbulence and free shear turbulence	60
5.5	Discussion	62
5.5.1	Why does attention work ?	62
5.5.2	Computational efficiency	65
5.6	Conclusion	65
CHAPTER 6 ARTICLE 3: LINEAR ATTENTION COUPLED FOURIER NEURAL OPERATOR FOR SIMULATION OF THREE-DIMENSIONAL TURBULENCE		
	67	
6.1	Introduction	67
6.2	The Fourier neural operator	71
6.3	Attention enhanced neural network	72
6.3.1	Self-attention coupled FNO	72
6.3.2	Linear attention approximation	75
6.4	Dataset description	76

6.5	Hyperparameters of neural networks	79
6.5.1	Input time steps T	79
6.5.2	Project dimension k	81
6.6	Performance benchmark	81
6.7	Computational efficiency	85
6.8	Conclusion	86
CHAPTER 7 GENERAL DISCUSSION		88
CHAPTER 8 CONCLUSION		90
8.1	Summary of works	90
8.2	Objective review	91
8.3	Limits and future research	91
REFERENCES		93

LIST OF TABLES

Table 4.1	Performance benchmarks of different models.	31
Table 4.2	Correlation (between the learned weights and the panel length) of different airfoils and prediction error.	38
Table 5.1	Prediction errors of neural network models at different Reynolds numbers	51
Table 5.2	Computational efficiency comparison of three different numerical approaches.	65
Table 6.1	Prediction errors at different input time steps T	80
Table 6.2	Prediction errors at different project dimension k	81
Table 6.3	Computational efficiency comparison of three different numerical approaches.	86

LIST OF FIGURES

Figure 1.1	Airfoil geometry	2
Figure 1.2	Airfoil in free stream	3
Figure 1.3	Instantaneous vorticity field at 20 time steps	4
Figure 1.4	Turbulence prediction task	4
Figure 1.5	Visualization of ten random 3D vorticity field	5
Figure 2.1	Comparison of a CNN (left) and a fully connected network (right). . .	10
Figure 2.2	Example of 1×1 convolution	11
Figure 2.3	AeroCNN Framework. Adopted from [1].	12
Figure 2.4	ESCNN architecture.	13
Figure 2.5	The Fourier neural operator architecture	15
Figure 2.6	The Fourier layer.	16
Figure 2.7	Self-attention module for image generation [2].	16
Figure 2.8	Attention enhanced Fourier neural operator (FNO).	17
Figure 2.9	Self-attention block.	18
Figure 2.10	Standard self-attention module architecture.	21
Figure 2.11	Linear self-attention module architecture.	24
Figure 2.12	Linear attention coupled Fourier neural operator (LAFNO).	24
Figure 4.1	ESCNN architecture.	31
Figure 4.2	Prediction performance on validation airfoils.	32
Figure 4.3	ESCNN training learning curve.	32
Figure 4.4	Fully connected layer evolution in training.	33
Figure 4.5	Airfoil vortex panels in free stream.	34
Figure 4.6	Computed vortex distribution as a function of the number of panels. .	35
Figure 4.7	Comparison of the converged layer pattern with 160 panels vortex strength distribution.	35
Figure 4.8	Neuron values of the fully connected layer with different number of airfoil coordinates.	35
Figure 4.9	Fully connected layer under constrained activation.	36
Figure 4.10	Panel distribution over the airfoil surface.	39
Figure 4.11	Randomly sampled airfoils and their panel length distribution.	40
Figure 4.12	Comparison of learned weights and panel length distribution at the same scale	40
Figure 5.1	Fourier neural operator (FNO) architecture.	45

Figure 5.2	Self-attention block.	47
Figure 5.3	Attention enhanced Fourier neural operator (FNO).	47
Figure 5.4	Random sampled initial conditions from the training set and testing set.	49
Figure 5.5	Vorticity prediction and absolute error at selected time steps.	52
Figure 5.6	Averaged vorticity spectrum on 200 test samples.	52
Figure 5.7	PDF of normalized vorticity at different time steps.	53
Figure 5.8	Condition average error of vorticity at different time steps.	53
Figure 5.9	Relative error comparison at consequent time steps.	53
Figure 5.10	Vorticity prediction and absolute error at different Reynolds numbers.	54
Figure 5.11	Average spectrum at different Reynolds numbers.	55
Figure 5.12	PDF of normalized vorticity at different Reynolds numbers.	56
Figure 5.13	Condition average error of vorticity at different Reynolds numbers.	56
Figure 5.14	Average relative error of vorticity at different Reynolds numbers.	56
Figure 5.15	Generalization error at different Reynolds numbers (models trained on $Re = 1 \times 10^5$ dataset).	58
Figure 5.16	Vorticity prediction and absolute error at different Reynolds numbers. (models trained on $Re = 1 \times 10^5$ dataset).	58
Figure 5.17	Vorticity prediction and absolute error at different resolutions.	59
Figure 5.18	Average relative error at different resolutions.	60
Figure 5.19	Average spectrum of vorticity at different resolutions.	61
Figure 5.20	Vorticity prediction and absolute error at selected time steps (Taylor-Green turbulence).	63
Figure 5.21	Averaged vorticity spectrum on 200 test samples (Taylor-Green turbulence).	63
Figure 5.22	Vorticity prediction and absolute error at selected time steps (free shear turbulence).	64
Figure 5.23	Spatial-averaged relative error comparisons on 200 test samples at consequent time steps. (a) relative error of Taylor-Green turbulence (b) relative error of free shear turbulence (c) ratios of mean relative error.	64
Figure 5.24	Normalized absolute error versus normalized vorticity gradient.	66
Figure 5.25	Comparison of feature maps.	66
Figure 6.1	Fourier neural operator (FNO) architecture.	73
Figure 6.2	Attention enhanced Fourier neural operator.	74
Figure 6.3	Standard self-attention module architecture.	74
Figure 6.4	Linear self-attention module architecture.	77
Figure 6.5	Linear attention coupled Fourier neural operator (LAFNO).	77

Figure 6.6	Ten random sampled initial fields of vorticity magnitude from the training set and testing set.	80
Figure 6.7	Learning curve (testing error) of FNO and LAFNO	83
Figure 6.8	Isosurfaces of the normalized vorticity magnitude $\Omega/\Omega^{rms} = 1.5$ and relative error $\epsilon = 0.5$ at selected time steps.	83
Figure 6.9	Vorticity (2D slice) prediction and absolute error at selected time steps.	84
Figure 6.10	Relative error comparison at consequent time steps	84
Figure 6.11	Averaged velocity spectrum on 100 test samples.	85
Figure 6.12	PDFs of the normalized vorticity component ω_z at different time steps.	86

LIST OF SYMBOLS AND ACRONYMS

V_∞	Free Stream Velocity
ρ_∞	Free Stream Density
q_∞	Dynamic Pressure
$Mach$	Free Stream Mach number
α	Angle of Attack
Re	Free Stream Reynolds number
γ	Vortex Panel Strength
L	Lift Force
C_l	Aerodynamic Lift Coefficient
C_d	Aerodynamic Drag Coefficient
$D \subset \mathbb{R}^d$	The spatial domain for the PDE
$x \in D$	Points in the the spatial domain
$a \in \mathcal{A} = \left(D; \mathbb{R}^{d_a}\right)$	The input coefficient functions
$u \in \mathcal{U} = \left(D; \mathbb{R}^{d_u}\right)$	The target solution functions
D_j	The discretization of (a_j, u_j)
$G^\dagger : \mathcal{A} \rightarrow \mathcal{U}$	The operator mapping the coefficients to the solutions
μ	A probability measure where a_j sampled from
$v(x) \in \mathbb{R}^{d_v}$	The neural network representation of $u(x)$
d_a	Dimension of the input $a(x)$
d_u	Dimension of the input $u(x)$
d_v	The dimension of the representation $v(x)$
$\kappa : \mathbb{R}^{2(d+1)} \rightarrow \mathbb{R}^{d_v \times d_v}$	The kernel maps $(x, y, a(x), a(y))$ to a $d_v \times d_v$ matrix
ϕ	The parameters of the kernel network κ
σ	the activation function
$\mathcal{F}$	Fourier transformation
$\mathcal{F}^{-1}$	inverse Fourier transformation
R	The linear transformation applied on the lower Fourier modes
W	The linear transformation applied on the spatial domain
k	Fourier modes (wave numbers)
k_{max}	The max Fourier modes used in the Fourier layer
N	The number of training pairs
n	The size of discretization
s	The resolution of the discretization

ν	The viscosity
T	The time interval $[0, T]$ for time-dependent equation
VPM	Vortex Panel Method
2D	Two-Dimensional
3D	Three-Dimensional
CFD	Computational Fluid Dynamics
PDF	Probability Density Function
ML	Machine Learning
NN	Neural Network
DNN	Deep Neural Network
CNN	Convolution Neural Network
RNN	Recurrent Neural Network
PINN	Physics-Informed Neural Networks
ESCNN	Element Spatial Convolution Neural Network
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
TF-Net	Turbulent-Flow Network
FNO	Fourier Neural Operator
FNO + Attention	Attention-enhanced Fourier Neural Operator
GPU	Graphics Processing Unit

CHAPTER 1 INTRODUCTION

Simulating the dynamics and the interactions of the fluids (including liquids and gases) is one of the most important problems in science and engineering. The fundamental basis of fluid dynamics is the Navier–Stokes equations [3]. The Navier–Stokes equations are Partial Differential Equations (PDEs) that describe the motion of fluid substances. They mathematically express conservation of momentum and conservation of mass for incompressible fluids, as shown in equation 1.1. The solution of these equations is the flow velocity $\mathbf{u}$, $\mathbf{u}$ is a vector field: to every point in a fluid, at any moment in a time interval, it gives a vector whose direction and magnitude are those of the velocity of the fluid at that point in space and at that moment in time [3]. Here ∇ is the Nabla operator, ρ is the flow density, t is time, p is the pressure, ν is the kinematic viscosity – a measure of a fluid’s internal resistance to flow under gravitational forces, $\mathbf{F}$ is the external force.

$$\begin{aligned}\nabla \cdot \mathbf{u} &= 0 \\ \rho \frac{\partial \mathbf{u}}{\partial t} &= -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F}\end{aligned}\tag{1.1}$$

Unfortunately, solving these PDEs can be very hard [3]. They rely on discretizing the space into a very fine mesh, which can be slow and inefficient [4].

The rising of deep learning techniques has opened new opportunities to boost the computational efficiency [5]. Neural Networks (NNs) have shown themselves to be very successful at approximating nonlinear functions [6]. Such strong approximation capacity makes NNs an ideal tool for approximating the fluid dynamics that involves the modeling of high-dimensional and highly non-linear complex dynamical systems. As Kutz states in his work [5], it’s only a matter of time before Deep Neural Networks (DNNs) play an important role in the fluids community. The potential of using NNs to accelerate fluid simulations has lately gained increasing attention [7–12]. Recent efforts have been made to approximate the entire Navier-Stokes equations by DNNs [13–16]. Once trained, these neural network models can make inference within seconds on modern computers, thus can be extremely efficient when compared with traditional solvers [17]. However, most neural network models suffer from the prediction accuracy issue [5, 15, 16]. Moreover, as the fluid becomes more inertial, the dynamic system becomes more non-linear, making accurate prediction becomes even more challenging [8, 17, 18].

The main purpose of this thesis is to improve the prediction accuracy over existing state of

the art neural network models for incompressible fluid simulations. We cover three typical kinds of fluid simulations problems with increasing level of complexity: (1) Two-dimensional inviscid flow, where no viscosity is involved. (2) Two-dimensional viscous turbulent flow. (3) Three-dimensional viscous turbulent flow. Note that the fluid dynamic system becomes more non-linear as the fluid becomes more turbulent, and the system becomes more complex as the degrees of freedom increase dramatically from two-dimensional flow to three-dimensional flow.

1.1 Two-dimensional inviscid flow

In chapter 4 we start with the two-dimensional inviscid flow, where no viscosity is involved. In this scenario the Navier-Stokes equations can be simplified to linear equations [3]. The problem of interest here is to calculate the lift coefficient of an airfoil in the flow, given the airfoil geometry and the flow direction.

An airfoil is the cross-sectional shape of a wing, an airfoil in moving fluid produces aerodynamic forces such as the lift force L . The airfoil geometry is represented by a sequence of points on the airfoil surface, where the points' coordinates are represented in the 2D Cartesian coordinate system as (x, y) , as shown in figure 1.1. The panels of an airfoil refer to the segments between two sequential points, as shown in figure 1.1. The panels are serially numbered from 1 to n according to their locations on the airfoil surface (starting from the trailing edge, along the upper surface to the leading edge and back around the lower surface to trailing edge).

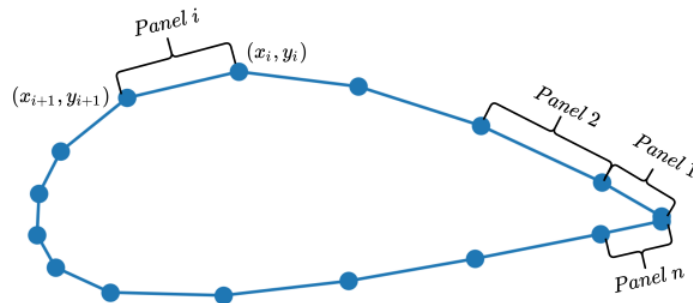


Figure 1.1 Airfoil geometry

Figure 1.2 illustrates the example of an airfoil in free stream, the free stream velocity is noted

by V_∞ . $Mach$ is the free stream velocity in Mach number. Re is the Reynolds number [3]: the ratio of inertial forces to viscous forces within a fluid that is subjected to relative internal movement due to different fluid velocities. The Reynolds number is defined by equation 1.2, where ρ is the density of fluids (kg/m^3), u is the flow speed (m/s), L is the characteristic linear dimension (m), μ is the dynamic viscosity of the fluid ($kg/(m \cdot s)$), ν is the kinematic viscosity of the fluid (m^2/s).

$$Re = \frac{uL}{\nu} = \frac{\rho uL}{\mu} \quad (1.2)$$

The angle of attack α is the angle between the free stream flow and the airfoil chord line, where the chord line is the straight line that connects the airfoil leading edge and trailing edge.

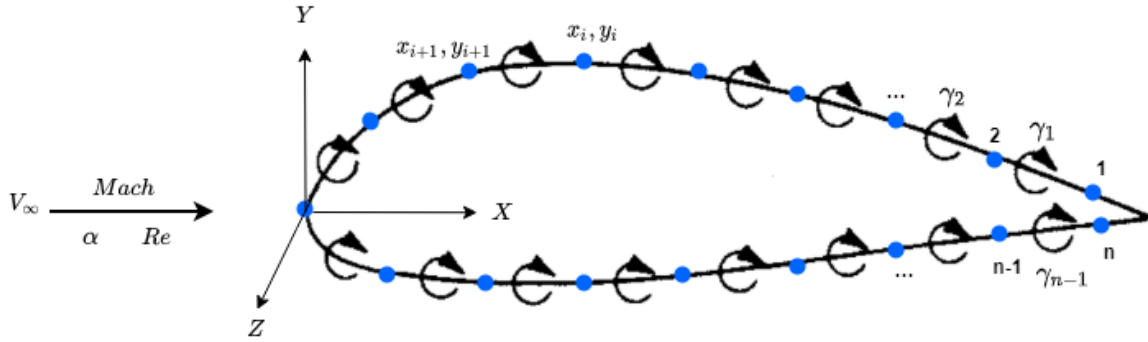


Figure 1.2 Airfoil in free stream

The task here is to calculate the lift coefficient C_l , C_l can be computed numerically via the Vortex Panel Method [3]. The VPM discretizes the surface of an airfoil by a series of sequential panels, where each panel yields a linear equation. Then the lift coefficient is obtained by solving the linear system [3].

Recently, Yao et al. proposed to apply the Convolution Neural Network (CNN) to predict the lift coefficient [1]. In this work, they processed the airfoils into images then fed the images as input into the CNN, and they reached the state of the art prediction accuracy.

1.2 Two-dimensional viscous flow

In chapter 5 we move on to a more complex problem: the two-dimensional (2D) viscous flow. In this problem, the viscosity is involved, thus the Navier-Stokes equations shown in equation

1.1 can no longer be simplified to linear equations, and can only be solved numerically via numerical solvers such as finite element methods or finite difference methods. The objective of solving the Navier-Stokes equations is to obtain the solution at each time step, where the solution refers to the velocity field $\mathbf{u}$ at each time step. In 2D viscous flow, the instantaneous velocity field $\mathbf{u}$ is represented by a 2D array. On the other hand, researchers tend to use the vorticity field ω for visualization, where $\omega = \nabla \times \mathbf{u}$. Figure 1.3 visualizes the evolution of the instantaneous vorticity field ω over 20 time steps, where the vorticity field of each time step is represented by a 64×64 tensor (same shape with the velocity field $\mathbf{u}$).

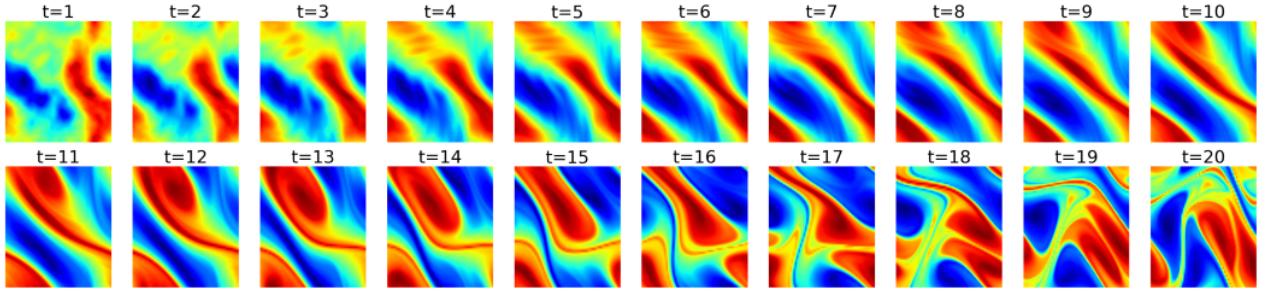


Figure 1.3 Instantaneous vorticity field at 20 time steps

However, the traditional Direct Numerical Simulation (DNS) approach relies on discretizing the space into a very fine mesh, which can be slow and inefficient. Recently, significantly efforts have been made to apply the NNs to accelerate the turbulence simulation [19–22]. Most of them aim to establish the input-output mapping with NNs, where the input is the solutions (velocity field or vorticity field) of previous T time steps, and the output is the solution of step $T + 1$, as shown in figure 1.4.

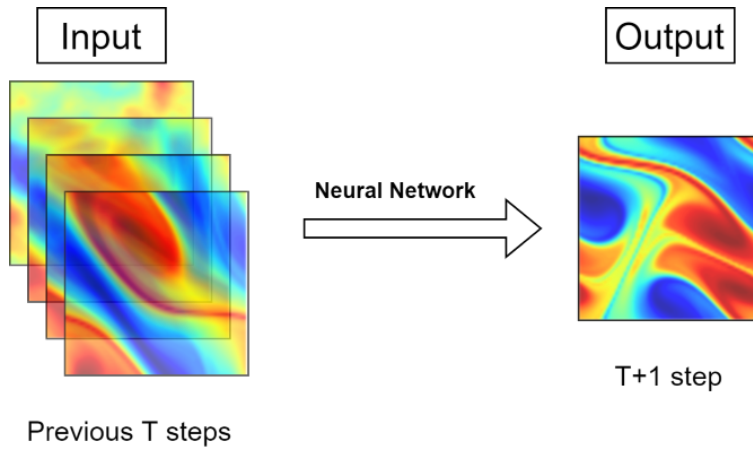


Figure 1.4 Turbulence prediction task

This task is very similar to the image-to-image mapping in computer vision, where the input and output are both arrays. Such an approach significantly improves the computational efficiency by avoiding solving the Navier-Stokes equations [19].

1.3 Three-dimensional viscous flow

In the case of a typical flow field of grid size 64, the instantaneous 2D vorticity field is of dimension 64×64 , whereas the instantaneous 3D vorticity field is of dimension $64 \times 64 \times 64 \times 3$, since the vorticity has three components $\omega_x, \omega_y, \omega_z$. Figure 1.5 visualizes ten random 3D vorticity field of grid size 64, where $\Omega = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$.

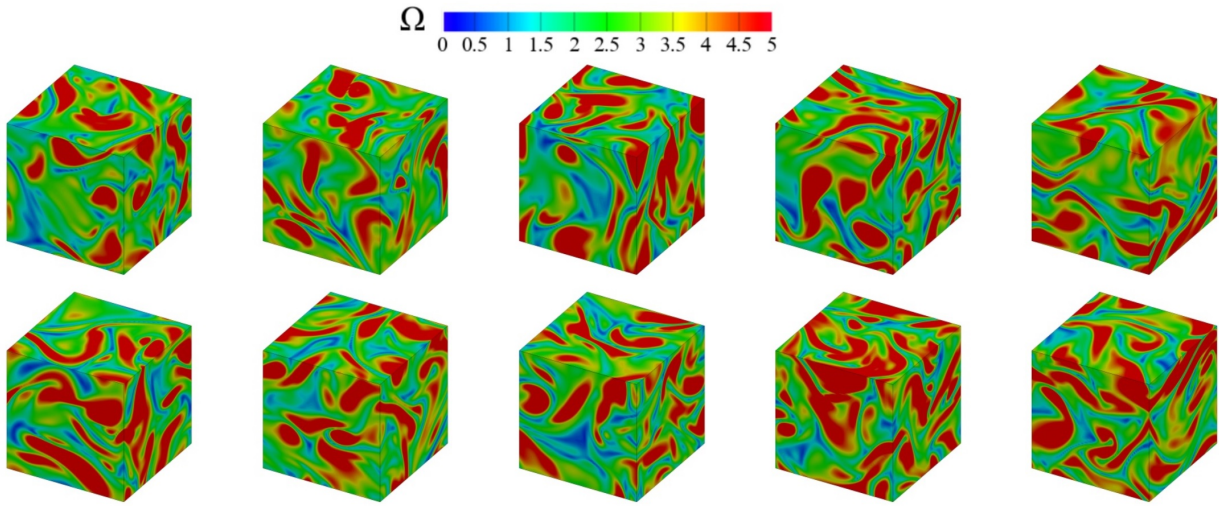


Figure 1.5 Visualization of ten random 3D vorticity field

In chapter 6 we focus on the three-dimensional viscous flow. Modeling the three-dimensional turbulence with deep neural networks is more challenging. The size and dimension of simulation data increases dramatically from 2D to 3D [23,24]. 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 [25]. Training such models can be computationally expensive because of the sheer amount of parameters involved [26]. Further, 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.

Recently, researchers have proposed neural network models for 3D turbulence simulation [27,28]. However, the prediction accuracy is still not satisfactory. The attention mechanism has shown itself to be very successful at boosting the performance of neural networks for

turbulence simulations, and therefore bringing new opportunities to improve the prediction accuracy of 3D turbulence simulation.

However, extending the attention mechanism from 2D 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 [29]. In the case of a typical 3D flow field of grid size $64 \times 64 \times 64$, computing the attention mapping requires 2034 GB memory for 32-bit floating point data type. Training and deploying the attention module can be prohibitively costly for 3D turbulence data, especially on GPUs where the available memory is constrained. Such quadratic complexity has become the main bottleneck for attention to be applied on 3D turbulence simulations.

Recently, Wang et al. demonstrated that the standard self-attention matrix can be approximated by a low-rank matrix [30]. 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 [30]. The linear attention approximation performs on par with standard self-attention, while being much more memory and time efficient [30], allowing the attention mechanism to be coupled with neural networks and trained on GPUs for 3D turbulence problem. In chapter 6, we couple the linear attention module with the Fourier neural operator to improve the prediction accuracy for 3D turbulence simulations.

Altogether, this dissertation explores neural network methods in fluid dynamic simulations. We propose neural network models to improve prediction accuracy over SOTA models in different simulation tasks.

1.4 Data description and general methodology

In the two-dimensional inviscid flow case, the airfoil data samples are taken from the database of UIUC Applied Aerodynamic Group [31], which covers a wide range of airfoil types from real-world designs. Each airfoil is represented by 160 points in x, y format, where the coordinates are interpolated using B-spline interpolation approach [32]. For the flow conditions, the Mach number $Mach$ and Reynolds number Re are fixed to 0.3 and 1.3×10^7 respectively, and the angle of attack α varies from -2° to 10° . The ground truth of C_l is computed with the CFD solver Xfoil [33], for each angle of attack at the specified flow condition for each airfoil geometry. In total 15678 samples are generated to create the final dataset. The whole dataset is divided into training set and validation set at the ratio of 8:2.

In the two-dimensional viscous flow case, the dimensionless Navier-Stokes equations for a viscous, incompressible fluid on the unit torus is described by equation (1.3), where $u \in$

$C([0, T]; H_{\text{per}}^r((0, 1)^2; \mathbb{R}^2))$ for any $r > 0$ is the velocity field, $\omega = \nabla \times u$ is the vorticity, $\omega_0 \in L_{\text{per}}^2((0, 1)^2; \mathbb{R})$ is the initial vorticity, Re is the Reynolds number, and $f \in L_{\text{per}}^2((0, 1)^2; \mathbb{R})$ is the forcing function.

$$\begin{aligned} \partial_t \omega(x, t) + u(x, t) \cdot \nabla \omega(x, t) &= (1/Re) \Delta \omega(x, t) + f(x), \\ \nabla \cdot u(x, t) &= 0, \\ \omega(x, 0) &= \omega_0(x), \\ x \in (0, 1)^2, t &\in [0, T]. \end{aligned} \tag{1.3}$$

The initial condition $\omega_0(x)$ is generated according to $\omega_0 \sim \mu$ where $\mu = \mathcal{N}(0, 7^{3/2}(-\Delta + 49I)^{-2.5})$ with periodic boundary conditions [4]. The forcing is kept fixed $f(x) = 0.1 (\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$. Data are generated on the grid size of 256×256 and are downsampled to 64×64 . Time is advanced with the Crank-Nicolson scheme, where the time-step is set to be 10^{-4} , and the solution is recorded every $t = 1$ time units. In this two-dimensional viscous flow case, we generate 1200 pairs of input-output data with the numerical solver, where each sample contains 20 steps of solutions of a random initialized condition. The solution is recorded every $t = 1$ time units. We use 1000 samples for training and 200 samples for testing. The neural network model takes the vorticity at previous 10 time steps solutions as input, and gives the vorticity at the next time step as output. During training, the vorticity of first 10 steps $\omega|_{(0,1)^2 \times [0,10]}$ 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 next 10 steps $\omega|_{(0,1)^2 \times [11,20]}$, which are labeled as ground truth.

In the three-dimensional viscous flow case, the dimensionless Navier-Stokes equations in conservative form for the 3D incompressible turbulence are given by equation (1.4), where $\mathbf{u}$ is the velocity, p is the modified pressure divided by the constant density, Re is the Reynolds number, and $\mathbf{F}$ is the large-scale force to maintain the turbulence statistically stationary.

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 \\ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} + \mathbf{F} \end{aligned} \tag{1.4}$$

The initial velocity field $\mathbf{u}(t=0)$ is randomly generated by the Gaussian distribution in spectral space [34], details are given in section 6.4. 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 [34–36]. 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 [34]. 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, C is the number of vorticity components. The neural network models predict the flow field of next step as output $(H \times W \times D \times C)$. In the three-dimensional viscous flow case, $H = W = D = 64$ 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 generate 3000 pairs of input-output data, where each sample contains 15 steps of solutions of a random initialized condition. The solution is recorded every $t = 1$ time units. The neural network models take the vorticity of previous four time steps solutions as input, and gives the vorticity of the next time step as output. During training, the vorticity of previous 4 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 of 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. We use 2400 samples for training and 600 samples for testing.

CHAPTER 2 LITERATURE REVIEW

This section introduces the reader to the concepts and literature related to the work presented in this dissertation. The reader is assumed to be familiar with basic neural networks and the basic physical concepts of fluid dynamics .

2.1 Neural networks for fluid dynamics

In recent years, significant efforts have been conducted to apply (Neural Networks) NNs to fundamental science research. In the field of chemistry, Noe et al. created PauliNet, a deep learning wavefunction ansatz that achieves nearly exact solutions of the electronic Schrödinger equation [37]. In the field of mathematics, NNs have been shown to be able to aid mathematicians in discovering new conjectures and theorems [38]. In the domain of game theory, evidence has been found that human knowledge is acquired by the AlphaZero neural network as it trains on the game of chess [39]. In the field of biology, scientists from DeepMind designed the AlphaFold, a type of NN that can generate models of proteins far more accurate than any that have come before [40]. A diverse collection of intersections between deep learning and physics is presented in the review paper by Carleo et al.. These applications range from statistics and quantum physics to high energy and cosmology [41].

The fluid dynamics community is no exception. Neural networks offer a wealth of techniques such as dimension reduction , feature representation learning and functional approximation methods that is widely needed in fluid dynamics community [6, 7, 42, 43]. NNs have shown themselves to be very successful at approximating the non-linear functions [6]. Such strong approximation capacity makes NNs an idea tool for approximating the fluid dynamics that involves the modeling of high-dimensional and highly non-linear complex dynamical systems.

Brenner et al. presented a perspective on how NNs might advance fluid mechanics and discussed the strengths and limitations of using machine learning techniques for such problems [44]. Singh et al. suggested the possibility and potential of extending the applicability of data-driven turbulence models to more general flows [45]. As Kutz states in his review paper [5], it's only a matter of time before DNNs play an important role in the fluids community.

2.2 Convolution neural network

Convolution neural networks (CNNs) are used as a key module of neural networks throughout this thesis. Here we introduce the basic concepts of CNNs, and especially 1×1 convolution. CNNs have famously enjoyed an impressive performance in many image recognition tasks [46]. Images in real-world have lots of edges and stripes, so CNNs can capture them well with local convolution kernel [46]. An important reason for the success of CNNs is that they have been shown to be able to learn to extract high-level semantic information from natural images [46–48]. Another advantage is that CNNs can take 3D input volume and arrange its neurons in three dimensions (width, height, depth), whereas the fully connected network can not, as shown in figure 2.1. The convolution layer transforms the 3D input volume to a 3D output volume of neuron activations [49, 50], such features naturally fit in with the fluids data.

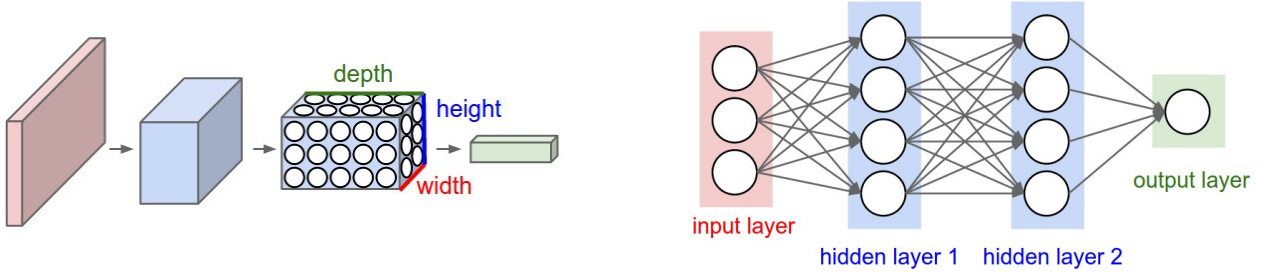


Figure 2.1 Comparison of a CNN (left) and a fully connected network (right).

In a convolution layer, the input volume is convolved with a kernel (or filter). The output of a convolution layer is also referred to as the convolved feature maps. A kernel refers to the sets of weights within a convolution layer. The kernel slides across the height and width of the input volume and the dot product of the kernel and the input are computed at every spatial position. The length by which the kernel slides is known as the stride length.

The kernel size F is the receptive field size of the convolution layer neurons. The spatial size of the output volume can be computed as a function of the input volume size $(W - F + 2P)/S + 1$, where W is the input volume size, F is the kernel size, P is the amount of zero padding on the border, and S is the stride [46]. For a convolution layer without padding, the spatial size of the output volume is usually smaller than the input volume size.

However, when $F = 1$, $P = 0$ and $S = 1$, the input volume size are kept the same after convolution, and such operation is called 1×1 convolution. 1×1 convolution is used in throughout this thesis. The concept of 1×1 convolution is firstly introduced by Lin et al. to enhance the performance of CNNs for image recognition task [51]. The most important

advantage of 1×1 convolution is that it allows information exchange across channels (the number of channels correspond to depth in figure 2.1), while keeping the height and width unchanged. Figure 2.2 shows an example of dimension reduction via the 1×1 convolution, the convolution kernel takes input volume of shape $64 \times 64 \times 192$ and output volume of shape $64 \times 64 \times 1$.

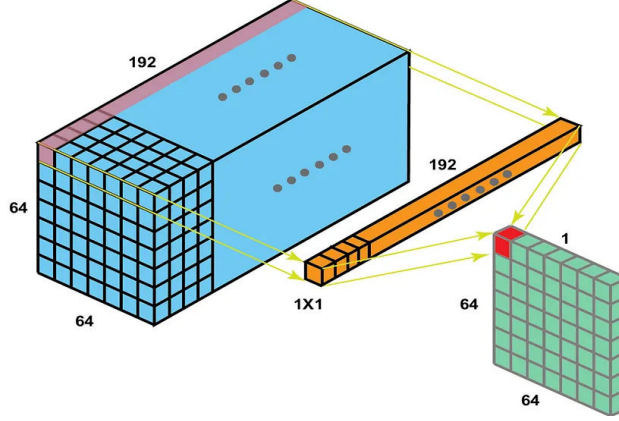


Figure 2.2 Example of 1×1 convolution

2.3 Predicting the airfoil lift coefficient in the two-dimensional inviscid flow

In chapter 4 we apply neural networks on the two-dimensional inviscid flow.

One of the most crucial tasks in the two-dimensional inviscid flow simulation is to predict the lift coefficient [52, 53]. Rai et al. adopt the ‘vanilla’ multi-layered perceptron (MLP) architecture for the lift coefficient prediction task [54], they demonstrated the computational efficiency advantage of neural networks compared with traditional CFD approaches. However, the number of MLP parameters grows exponentially [6]. Furthermore, the MLPs are not capable of extracting the spatial correlations between input dimensions, whereas the spatial information is important in CFD modeling. On the other hand, CNNs are able to learn the local connectivity features and the spatial correlations [55, 56].

CNNs have shown themselves to be very successful at classifying the images. Is it possible that CNNs can also be extended to predict the airfoil lift coefficient? Yao et al. proposed the AeroCNN-II (AeroCNN for short) to successfully illustrate this concept of the methodology [1]. In the AeroCNN framework, the airfoils are processed into images and fed as input to the CNN. Figure 2.3 illustrates the AeroCNN framework [1].

The AeroCNN is modified on top of a standard CNN architecture, the LeNet [57], which has been shown to be very effective on the handwritten digit recognition task. The AeroCNN

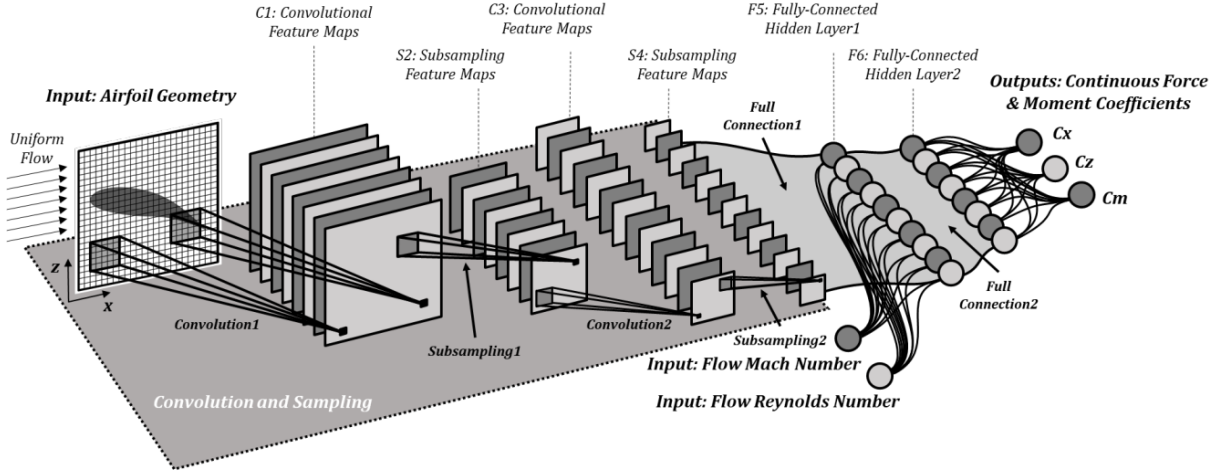


Figure 2.3 AeroCNN Framework. Adopted from [1].

framework is the first of its kind to investigate the 2D aerodynamic problems involving diverse flow conditions and the variety of sectional shapes in a single framework of the CNN using the concept of the “artificial image” [1]. The AeroCNN achieved state-of-the-art prediction accuracy.

In chapter 4 we further improve the prediction accuracy by proposing a novel architecture, the Element Spatial Convolution Neural Network (ESCNN). ESCNN is an end-to-end neural network that takes the airfoil coordinates x_j, y_j and angle of attack α as input, and gives the lift coefficient C_l as output. The ESCNN architecture is straightforward. As shown in figure 2.4, it consists of two standard convolution layers followed by non-linear activation function, and a fully connected layer before the prediction [58]. The two convolution layers use 200 convolution filters, where the filter size of first layer is 5×1 and the second layer is 1×1 respectively. Note that since the ESCNN takes sequential airfoil coordinates as input, and only convolutions and activations are involved in the architecture, therefore all the hidden layers are not permutation invariant.

The ESCNN adopts a physics-informed structured input, and it achieves over 60% error reduction compared with the state-of-the-art AeroCNN model. Furthermore, the parameter of ESCNN is two orders of magnitude smaller than AeroCNN, thus can be more computational efficiency compared with the AeroCNN.

In addition to accuracy, we also investigated the interpretability issue. In Chapter 4, we show that the ESCNN has the ability to extract physical patterns that emerge from aerodynamics, and such patterns are clearly reflected within a layer of the network. We further show that

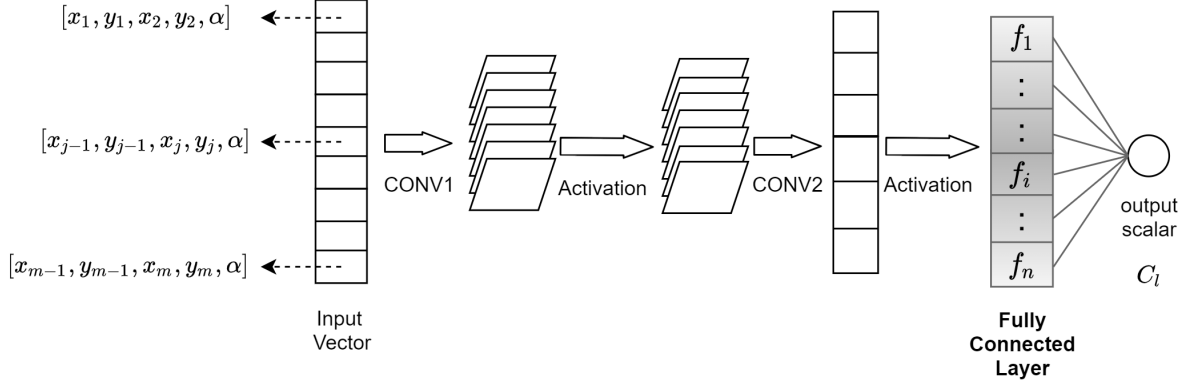


Figure 2.4 ESCNN architecture.

the ESCNN is capable of learning the physical laws and equation of aerodynamics from simulation data.

2.4 Two-dimensional turbulent flow prediction with NNs

In Chapter 5 we focus on the two-dimensional turbulent flow, which is more complex than the two-dimensional inviscid flow.

Over the past few years, significantly efforts have been made in recent years to accelerate the simulation of turbulent flow with neural networks [7,8]. Most of them aim to approximate the entire Navier-Stokes equations by deep neural networks [13–16]. Such approaches improve the computational efficiency by avoiding solving the Navier-Stokes equations [19–22]. Once trained, the "black-box" neural network models can make inference within seconds on modern computers, thus can be extremely efficient when compared with traditional approaches [17].

Jin et al. applied NNs to solve the Navier-Stokes equations [59]. Kochkov et al. applied NNs to accelerate Computational Fluid Dynamics (CFD) simulations, and achieved a significant reduction in computation cost [60]. Ren et al. proposed a deep learning framework that combines the long short-term memory network and the convolution neural network for prediction of temporal evolution of turbulent flames [61]. Jin et al. proposed a convolutional neural networks using measurements of the pressure field on the cylinder to predict the velocity field around a cylinder [62]. Cheng et al. used the deep residual neural network to predict the flow velocity fields [63]. 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) [64]. Despite that these models are computational efficient, they fail to reach the desired prediction accuracy [18,47,65].

2.4.1 The Fourier neural operator

Recently, Li et al. proposed the Fourier neural operator (FNO) to learn the Navier-Stokes equation [4]. The FNO achieves 1% error rate on 2D turbulence prediction task at low Reynolds numbers [4]. Experiments show that the FNO achieves state-of-the-art prediction accuracy [4]. In chapter 5 and chapter 6, we further improve the prediction accuracy over the state-of-the-art FNO model.

The main difference between most existing NNs and the FNO is that the FNO is parameterized in Fourier space. Since the inputs and outputs of PDEs are continuous functions, it is more efficient to represent them in Fourier space. The overall architecture of Fourier Neural Operator (FNO) is shown in figure 2.5. In the 2D turbulence simulation task, the FNO takes input tensor $a(x)$ of shape $64 \times 64 \times 10$, where the first two dimension 64×64 is the grid mesh in the spatial x, y direction, and the last dimension 10 is the number of sequential time steps [4].

First, the input $a(x)$ is lifted to a higher dimension channel space by a fully connected layer P , where P is a fully connected layer parameterized by a 10×20 real-valued weight matrix. Then a sequence of Fourier layers is applied, where the Fourier layer keeps the tensor shape unchanged. Finally the output $u(x)$ is obtained by projecting back to the target dimension by another fully connected layer Q , where Q is a fully connected layer parameterized by a 20×1 real-valued weight matrix.

The core of FNO is the sequence of Fourier layers [4], as shown in figure 2.6. The Fourier layer adopts an encoder-decoder structure, and it mainly consists of three operations: 1. Fourier transform $\mathcal{F}$; 2. linear transform on the truncated Fourier modes R ; 3. inverse Fourier transform $\mathcal{F}^{-1}$.

The Fourier layer takes $v_i \in \mathbb{R}^{64 \times 64 \times 20}$ as input tensor, and v_i is permuted into $20 \times 64 \times 64$ before the Fourier transformation. Thus $v_i \in \mathbb{R}^{20 \times 64 \times 64}$ and $\mathcal{F}(v_i) \in \mathbb{C}^{20 \times 64 \times 64}$, then the higher modes are truncated to obtain $\mathcal{F}(v_i) \in \mathbb{C}^{20 \times 12 \times 12}$ [4]. The weight matrix R is parameterized by a $20 \times 12 \times 12$ complex-valued tensor, thus $\mathcal{F}(v_i)R \in \mathbb{C}^{20 \times 12 \times 12}$, and $\mathcal{F}(v_i)R$ is zero-padded into $20 \times 64 \times 64$ complex-valued matrix before the inverse Fourier transform $\mathcal{F}^{-1}$. The non-linear activation is applied after the inverse Fourier transform, thus $v_{i+1} = \sigma(\mathcal{F}^{-1}(\mathcal{F}(v_i)R) + Wv_i)$.

However, when the fluid dynamic system becomes more chaotic and complex at higher Reynolds numbers, the FNO suffers from prediction accuracy drop issue [4]: the error rate arises over 15% at the Reynolds number of 10^5 . In chapter 5, we resolve this issue with the attention mechanism. We couple the self-attention module with FNO to improve the

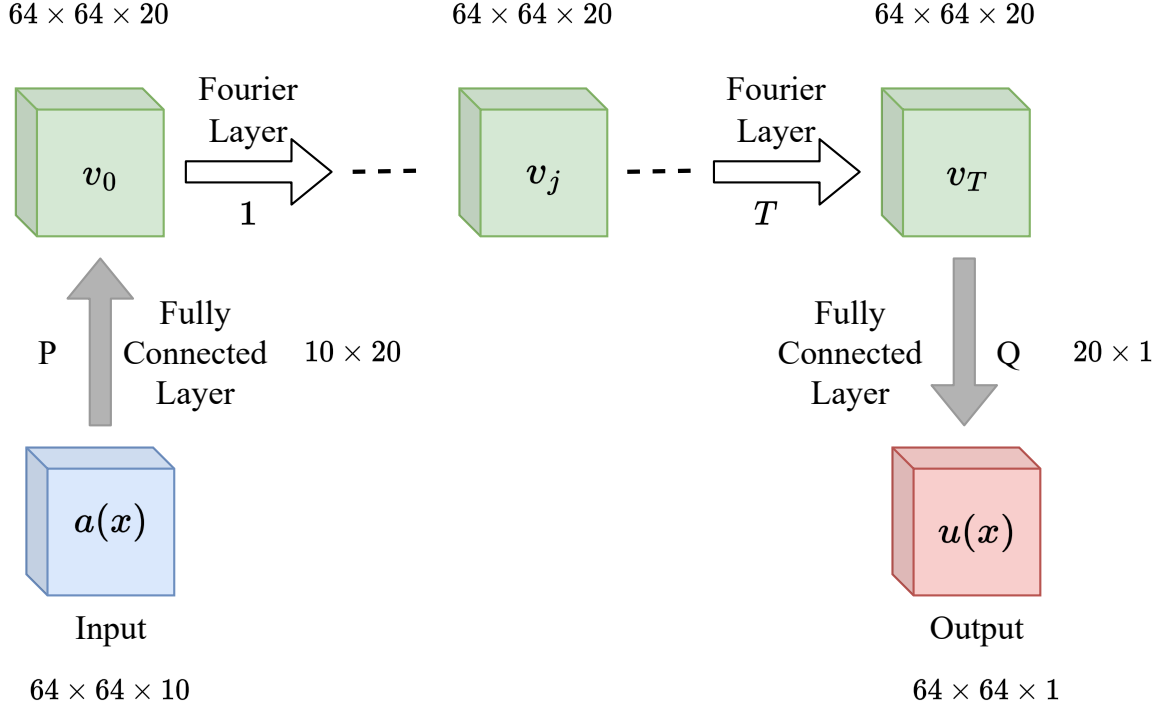


Figure 2.5 The Fourier neural operator architecture

prediction accuracy of FNO.

2.4.2 Self-attention mechanism

In recent years, attention mechanism has been widely used to boost the performance of neural networks on a variety of tasks including Natural Language Processing (NLP) [66], Speech Recognition [67] and Computer Vision [68].

Mnih et al. first proposed to apply the visual attention for image classification task, the intuition is to focus on relevant regions within the image, and capture structural long-range dependencies between parts of the image [69]. Visual attention also provide significant improvement for object detection task, since it can help to recognize and localize the target objects within the image [70]. Gregor et al. proposed the Deep Recurrent Attentive Writer (DRAW) to apply visual attention for the image generation task [71]. DRAW adopts a step by step encoder-decoder framework for image generation. The visual attention is used to selectively attend to certain regions of the input image while regenerating corresponding scenes within the image in an iterative manner.

The self-attention mechanism is firstly introduced by Zhang et al. to improve the performance

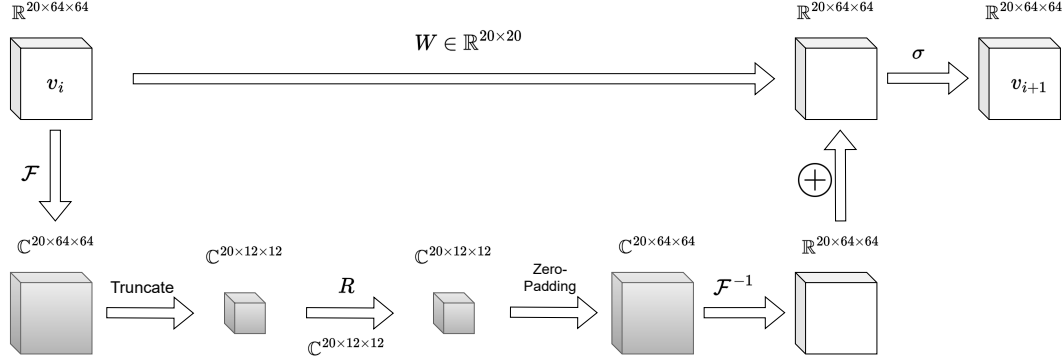


Figure 2.6 The Fourier layer.

of CNNs on the image generation task [2]. Convolution processes the information in a local neighborhood, thus using convolution layers alone is computationally inefficient for modeling long-range dependencies in images [72–74]. In the research by Zhang et al., the self-attention module is used to calculate the response at a position as a weighted sum of the features at all positions, which enables the neural network to efficiently model relationships between widely separated spatial regions [2].

The self-attention module is shown in figure 2.7, where the $\otimes$ denotes matrix multiplication and the softmax operation is performed on each row.

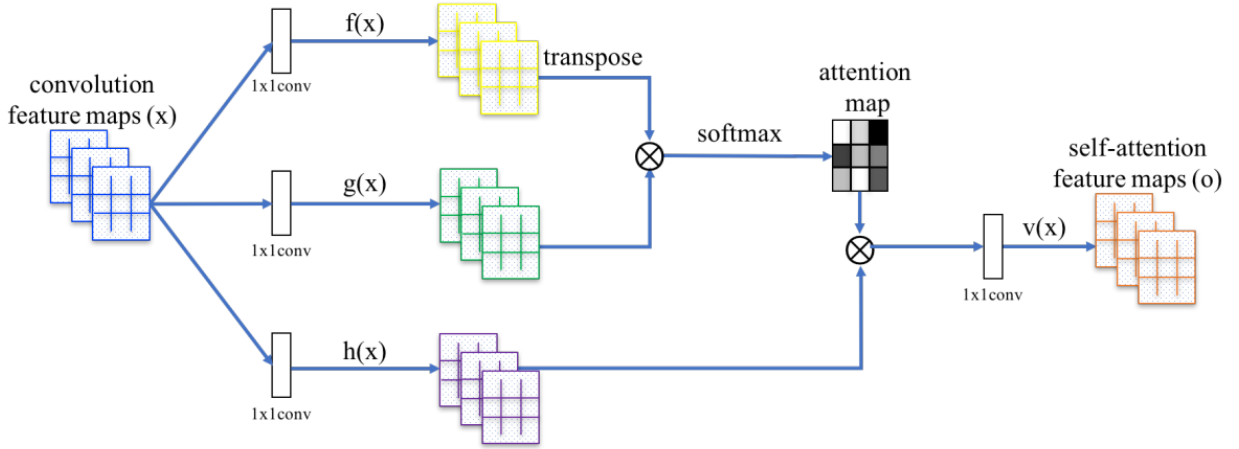


Figure 2.7 Self-attention module for image generation [2].

The self-attention module is applied after each convolution layer. The image features from the previous convolution layer $x \in \mathbb{R}^{C \times N}$ are first transformed into two feature spaces f, g to calculate the attention, where $f(x) = W_f x$ and $g(x) = W_g x$. Here, C is the number of channels and N is the number of feature locations of features from the previous convolution

layer. The elements of attention map $\beta_{j,i}$ is calculated by equation 2.1, and $\beta_{j,i}$ indicates the extent to which the model attends to the i^{th} location when synthesizing the j^{th} region.

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^N \exp(s_{ij})}, \text{ where } s_{ij} = \mathbf{f}(\mathbf{x}_i)^T \mathbf{g}(\mathbf{x}_j) \quad (2.1)$$

The output of the attention layer (self-attention feature maps) is $\mathbf{o} = (\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_j, \dots, \mathbf{o}_N) \in \mathbb{R}^{C \times N}$, where $\mathbf{o}_j$ is given by equation 2.2.

$$\mathbf{o}_j = \mathbf{v} \left(\sum_{i=1}^N \beta_{j,i} \mathbf{h}(\mathbf{x}_i) \right), \mathbf{h}(\mathbf{x}_i) = \mathbf{W}_h \mathbf{x}_i, \mathbf{v}(\mathbf{x}_i) = \mathbf{W}_v \mathbf{x}_i \quad (2.2)$$

$\mathbf{W}_g \in \mathbb{R}^{\bar{C} \times C}$, $\mathbf{W}_f \in \mathbb{R}^{\bar{C} \times C}$, $\mathbf{W}_h \in \mathbb{R}^{\bar{C} \times C}$, and $\mathbf{W}_v \in \mathbb{R}^{C \times \bar{C}}$ are learned weights and are implemented as 1×1 convolutions, where $\bar{C} = C/8$ is a common experimental setting. The self-attention mechanism has been shown to be effective in modeling long-range dependencies [2].

2.4.3 Self-attention enhanced FNO

In chapter 5 we couple the attention mechanism with the FNO, to improve the accuracy of FNO on two-dimensional turbulence prediction task.

The sequence of functions $v_0 \mapsto v_1 \mapsto \dots \mapsto v_T$ is the core structure of FNO. The Fourier layer in FNO can be viewed as a substitute for the convolution layer [4], and we use the self-attention block to refine the FNO output as feature augmentation. The self-attention block is implemented after the Fourier layer and before the fully connected layer Q , it takes v_t as input and output the attention refined feature maps v'_t , as shown in figure 2.8.

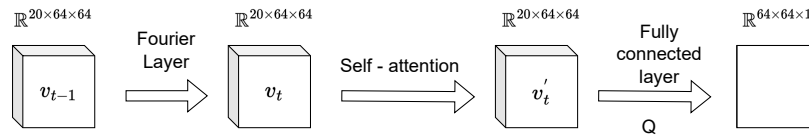


Figure 2.8 Attention enhanced Fourier neural operator (FNO).

The overall architecture of the self-attention block is shown in Fig.2.9, it learns sequential attention maps that are called the channel attention and the space attention. The first layer performs 1×1 convolutions on $v_t(\mathbf{x})$ to sequentially infer the channel attention map $\mathbf{M}\mathbf{c}$ and the space attention map $\mathbf{M}\mathbf{s}$, as shown in Eq.2.3.

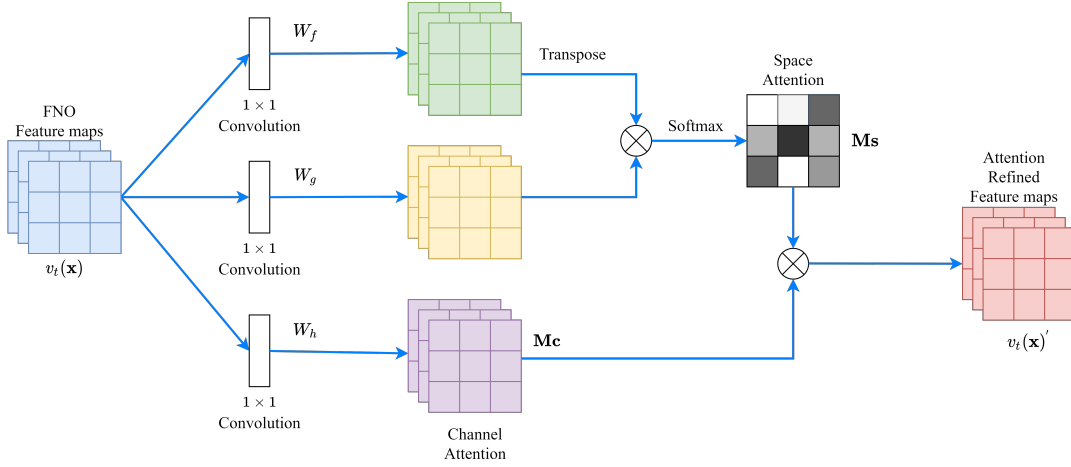


Figure 2.9 Self-attention block.

$$\mathbf{M}_c = W_h v_t(\mathbf{x}), \mathbf{M}_s = \frac{\exp(s_{ij})}{\sum_{i=1}^N \exp(s_{ij})}, \text{ where } s_{ij} = (W_f v_t(\mathbf{x}_i))^T (W_g v_t(\mathbf{x}_j)). \quad (2.3)$$

The convolution parameters W_f, W_g, W_h learn the embedding of query, key and value, respectively. Eq. 2.4 gives the attention refined feature maps, where $\otimes$ denotes the dot product.

$$v_t(\mathbf{x})' = \mathbf{M}_s \otimes \mathbf{M}_c. \quad (2.4)$$

Despite that the self-attention module allows feature refinement for any neural operators $v_t \Rightarrow v_t'$ for $t = 1, \dots, T$, our empirical experiments show that a single block at the end of sequence $v_T \Rightarrow v_T'$ has the same level of improvement as refining all neural operators in the sequence $v_1 \Rightarrow v_1' \mapsto \dots \mapsto v_T \Rightarrow v_T'$. The attention parameters W_f, W_g, W_h can be jointly learned with the Fourier layer during training. More importantly, since the attention block adopts 1×1 convolution filters, it retains the mesh invariant property of neural operators, meaning that the attention-enhanced FNO still keeps the ability of training on coarse grid and making inference on finer grid without seeing any higher resolution data. Details are discussed in Sec. 5.4.4.

In chapter 5 we show that the attention-enhanced FNO outperforms the original FNO, and can accurately reconstruct a variety of statistics and instantaneous spatial structures of turbulence at high Reynolds numbers flow. Furthermore, the attention mechanism provides 40% error reduction with 1% increase of parameters, at the same level of computational cost.

2.5 3D turbulence simulation with NNs

The attention mechanism has been shown to be very successful at boosting the neural networks performance for 2D turbulence simulations, in Chapter 6 we extend the attention mechanism from 2D to 3D turbulence applications.

2.5.1 Prediction accuracy issue

Compared with the spatial-temporal reconstruction of 2D turbulence [19,63,75,76], modeling the 3D turbulence with deep neural networks is much more challenging [23,24]. The size and dimension of simulation data increases dramatically from 2D to 3D. Moreover, 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 [26]. Training such models can be computationally expensive because of the sheer amount of parameters involved. Further, 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 [27]. 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 [27]. The CGANs model has an acceptable accuracy in modelling the velocity features of individual snapshots of the flow, but has difficulties in modelling the probability density functions (PDFs) of the passive scalars advected with the velocity [27].

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 [77]. 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 [77], 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 utilising the convolutional operators [27]. 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 [27]. The CC-LSTM is able to predict the spatial-temporal dynamics of flow: the model can accurately reconstruct the flow field in the large scale, but it misses details in the small scale.

Accurate prediction of 3D turbulence is still one of the most challenging problems for neural networks.

2.5.2 Self-attention quadratic complexity

The attention mechanism has been widely used in boosting the performance of neural networks on fluid simulations. Wu et al. introduced the self-attention into a convolution auto-encoder to extract temporal feature relationships from high-fidelity numerical solutions [78]. 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 [78]. 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 [79]. 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 [80]. Kissas et al. coupled the attention mechanism with the neural operators towards 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 [81].

The attention mechanism has been shown to be very successful at boosting the neural networks performance for 2D turbulence simulations, and therefore bringing new opportunities to improve the prediction accuracy for 3D turbulence simulation.

An attention function can be described as mapping a query and a set of key-value pairs to an output. 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 [29]. The architecture of the self-attention block is shown in Fig.2.10. 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. (2.5). The standard softmax function $\mathbb{R}^K \rightarrow (0, 1)^K$ is defined by the Eq. (2.6). In the 2D turbulence simulation task [82], the 2D turbulence data is generated on the grid size of 64×64 , such that $n = 64 \times 64$ and $d = 20$.

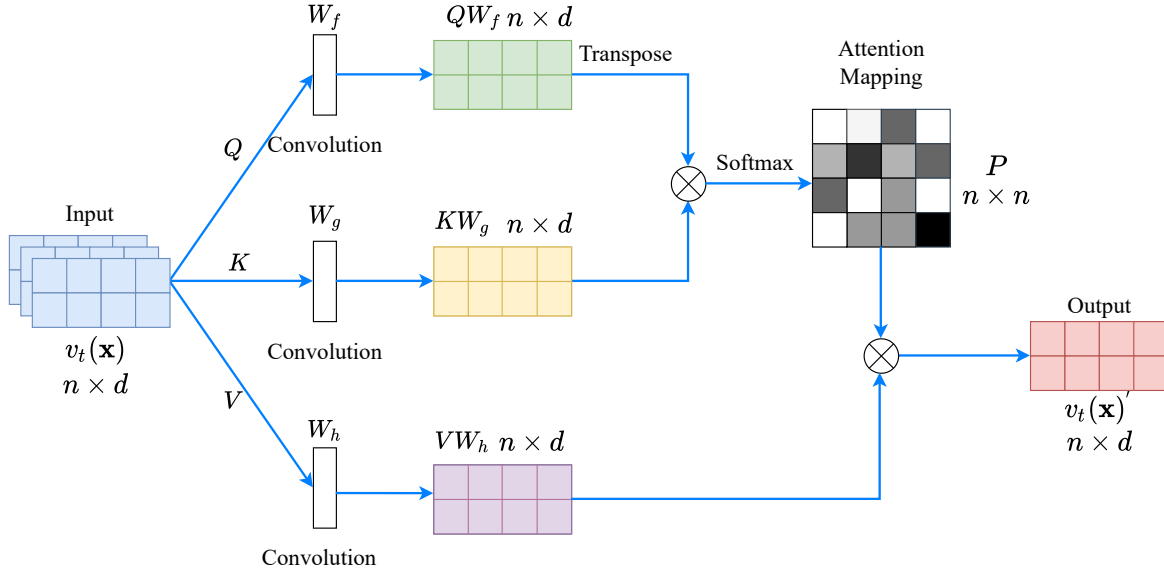


Figure 2.10 Standard self-attention module architecture.

$$v_t(\mathbf{x})' = \text{Attention}(QW_f, KW_g, VW_h) = \underbrace{\text{softmax}\left[\frac{QW_f (KW_g)^T}{\sqrt{d}}\right]}_P VW_h \quad (2.5)$$

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \text{for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K \quad (2.6)$$

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 [83–85]. It is noted from Fig. 2.10 and Eq. (2.5) 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 n becomes $n = 64 \times 64 \times 64 \times 3$, 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.

2.5.3 Linear self-attention coupled FNO for 3D turbulence

We resolve this bottleneck with the concept of linear attention. Recently, Wang et al. showed that the attention mapping matrix P can be approximated by a low-rank matrix $\tilde{P}$ [30]: 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

$$\Pr \left(\left\| \tilde{P}w^T - Pw^T \right\| < \epsilon' \left\| Pw^T \right\| \right) > 1 - o(1). \quad (2.7)$$

Here $\Pr$ refers to probability, ϵ' is a small constant, $o(1)$ 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 shown in equation 2.8, where σ_i , u_i and v_i are the i largest singular values and their corresponding singular vectors.

$$P \approx P_{low} = \sum_{i=1}^k \sigma_i u_i v_i^T = \underbrace{[u_1, \dots, u_k]}_k \text{diag} \{ \sigma_1, \dots, \sigma_k \} \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} \} k \quad (2.8)$$

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 [30]. 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 [30, 86].

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 [30], for any row vector w of $QW_f (KW_g)^T / \sqrt{d}$,

$$\Pr \left(\left\| \text{softmax} \left(w E^T \right) F V W_h - \text{softmax}(w) V W_h \right\| \leq \epsilon' \left\| \text{softmax}(w) \right\| \left\| V W_h \right\| \right) > 1 - o(1) \quad (2.9)$$

Detailed mathematical proof can be found in the reference [30].

The architecture of linear attention is shown in Fig. 2.11. 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 $(n \times k)$ -dimensional attention mapping matrix $\bar{P}$ using the scaled dot-product attention, as described by Eq. (2.10). 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 normal distribution $N(0, 1/k)$ and δ is a small constant [30].

$$\begin{aligned} \overline{v_t(\mathbf{x})'} &= \text{Attention}(QW_f, EKW_g, FVW_h) \\ &= \underbrace{\text{softmax}\left(\frac{QW_f (EKW_g)^T}{\sqrt{d}}\right)}_{\bar{P}: n \times k} \cdot \underbrace{FVW_h}_{k \times d}, \end{aligned} \quad (2.10)$$

The linear self-attention module performs on par with standard self-attention module [30], 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 Chapter 6, we replace the self-attention block by the linear attention module for 3D turbulence simulation task. The architecture of our proposed model, the linear attention coupled Fourier neural operator (LAFNO), is shown in Fig. 2.12. In our numerical simulation, the maximum GPU memory consumption of training LAFNO is 35.82 GB. Furthermore, numerical simulations show that the linear attention provides 40% error reduction at the same level of computational cost, compared with the original FNO.

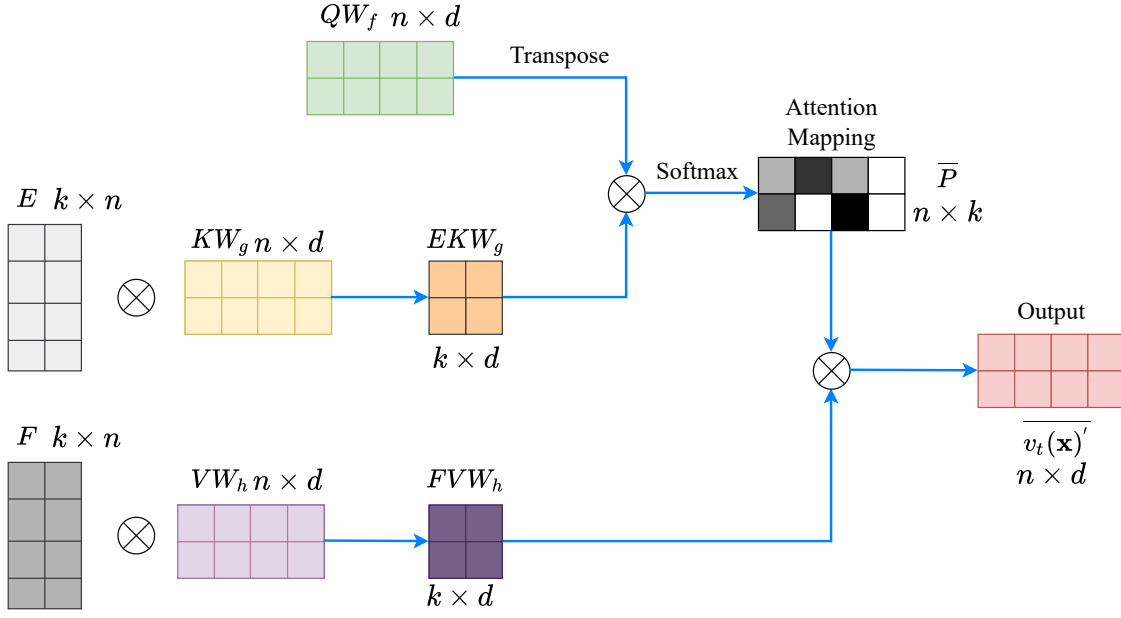


Figure 2.11 Linear self-attention module architecture.

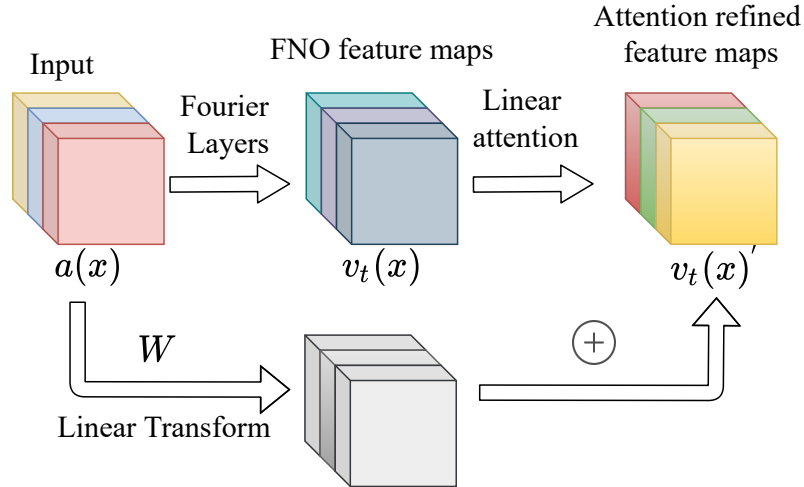


Figure 2.12 Linear attention coupled Fourier neural operator (LAFNO).

CHAPTER 3 OBJECTIVES AND CONTRIBUTIONS

This dissertation is mainly concerned with the improvement of prediction accuracy over existing state of the art neural network models for fluid simulations. We cover three typical kinds of fluid simulations problems with increasing level of complexity: (1) Two-dimensional inviscid flow, where no viscosity is involved. (2) Two-dimensional viscous turbulent flow. (3) Three-dimensional viscous turbulent flow.

For each kind of simulation problem, we present a neural network approach, and we compare the prediction accuracy with existing state-of-the-art model as baseline.

3.1 Structure of the dissertation

The specific contributions of this dissertation are summarized below, along with associated publications, for each chapter.

Chapter 4

This chapter has been published at the Scientific Reports journal.

Overall contribution

We propose a neural network model to improve the prediction accuracy of airfoil lift coefficient in two-dimensional inviscid flow.

Specific contributions

In the two-dimensional inviscid problem, we improve the prediction accuracy of airfoil lift coefficient by proposing a novel architecture, the Element Spatial Convolution Neural Network (ESCNN). The ESCNN adopts a physics-informed structured input, and it achieves over 60% error reduction compared with the state-of-the-art AeroCNN model. Moreover, the number of ESCNN parameters is two orders of magnitude smaller than AeroCNN, thus can be more computational efficient.

We further provide insight physical interpretations as to how ESCNN makes accurate predictions. We investigate a layer of the ESCNN during training. We discover that the ESCNN

has the ability to extract physical patterns that emerge from aerodynamics, and such patterns are clearly reflected within a layer of the network. We show that the ESCNN is capable of learning the physical principles of aerodynamics from simulation data.

Chapter 5

This chapter has been published at the Physics of Fluids journal.

Overall contribution

We couple the attention mechanism with FNO, to improve the performance of FNO in two-dimensional viscous turbulent flow prediction task.

Specific contributions

We improve the prediction accuracy of 2D turbulent flow using attention mechanism as augmentation to enhance the FNO performance.

We propose an attention-enhanced neural network model. We compare the model performance against the state-of-the-art neural network as the baseline, the Fourier neural operator (FNO), on a two dimensional turbulence prediction task. Numerical experiments show that the attention-enhanced neural network model outperforms existing state-of-the-art baselines, and can accurately reconstruct a variety of statistics and instantaneous spatial structures of turbulence at high Reynolds numbers. Furthermore, the attention mechanism provides 40% error reduction with 1% increase in parameters, at the same level of computational cost.

Chapter 6

This chapter has been accepted by the Physics of Fluids journal.

Overall contribution

We extend the attention mechanism from 2D to 3D turbulent flow prediction.

Specific contributions

We extend the self-attention mechanism to 3D turbulence data, to improve the prediction accuracy of 3D turbulence simulation.

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. We resolve this issue with the concept of linear attention. 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 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.

CHAPTER 4 ARTICLE 1: LEARNING AERODYNAMICS WITH NEURAL NETWORK

This Chapter has been published at the Scientific Reports Journal on April 26th 2022, authored by Wenhui Peng, Yao Zhang, Eric Laurendeau and Michel Desmarais.

4.1 Introduction

In recent years, significant efforts have been conducted to apply NNs to fundamental science research. Two representative works revolutionized fundamental scientific research. Noe et al. created PauliNet, a deep learning method that is capable of solving Schrödinger equation [37]. Scientists from DeepMind designed the AlphaFold, a type of NN that can generate models of proteins far more accurate than any that have come before [40]. A diverse collection of intersections between NNs and physics is presented in the review paper by Carleo et al.. These applications range from statistics and quantum physics to high energy and cosmology [41].

The fluid dynamics community is no exception. The potential of using NNs to tackle fluid mechanics problems has lately been gained increasingly attention [7, 8]. Jin et al. proposed NSFnets that applied NNs to solve the Navier-Stokes equations [59]. Kochkov et al. applied NNs to accelerate Computational Fluid Dynamics (CFD) simulations, and achieved a significant reduction in computation cost [60]. Li et al. further improved the computational efficiency by approximating the Navier-Stokes equations with the Fourier neural operator [4]. Once trained, these NN models can make inference within seconds and can be extremely efficient compared with traditional CFD approaches [13–17, 64, 87].

However, these NN models suffer from generalization problems: their prediction accuracy drops dramatically once the parameters of the partial differential equations change [60]. To resolve this issue, Raissi et al. proposed the Physics-Informed Neural Networks (PINNs). They developed a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations [88]. In the PINN framework, prior physical knowledge is introduced as constraint, such that NNs are trained to solve supervised learning tasks while respecting the given laws of physics described by general nonlinear partial differential equations [88]. With the physical constraints, the PINNs manage to achieve state-of-the-art prediction accuracy [59, 89–93].

Despite these significant improvements towards prediction accuracy, scientists are not fully convinced with the results of NNs due to the lack of interpretability. How do PINNs solve

physical problems? Do they make inference based on physical principles or just probability? Answering these questions is necessary, since people need interpretations to understand the logic driving the learned model, and make sure that the predicted results are trustworthy. However, interpreting the complex interactions between parameters and layer nodes has always been a challenging task [94–97]. This is due to the nature of NNs, also known as the "black-box" issue: instead of solving the tasks with logical steps, NNs learn by examples and adjust parameters to improve their performance over time [6].

Recent progress has been made to interpret how NNs solve scientific problems. In the field of mathematics, NNs have been shown to be able to aid mathematicians in discovering new conjectures and theorems [38]. In the field of chemistry, NNs have been shown to be able to capture the complex pattern of electrons moving around the nucleus [37]. In the domain of game theory, evidence has been found that human knowledge is acquired by the AlphaZero neural network as it trains on the game of chess [39].

Is it possible that the NNs are able to develop physical insight in the process of solving physical problems, just like in other scientific domains?

Motivated by this question, in this work, we pierce the “black box” of a NN and investigate how NN solves a physical problem in aerodynamics. Our contributions are as follow: 1). We proposed a NN model that adopts a novel physics-informed structured input, the ESCNN, it outperforms existing state-of-the-art NNs in the airfoil lift coefficient prediction task. 2). We provide insight explanations as to how the ESCNN succeeds in making accurate predictions with standard convolution layers.

4.2 Predicting lift coefficient with neural networks

Calculating the airfoil lift coefficient is one of the most critical tasks in aerodynamics. It is generally achieved by using traditional Computational Fluid Dynamics (CFD) methods, which are often known for being computational expensive. We propose a neural network based model: Element Spatial Convolution Neural Network (ESCNN), to efficiently predict the airfoil lift coefficient [58].

ESCNN is an end-to-end neural network that takes the airfoil coordinates x_j, y_j and angle of attack α as input, and gives the lift coefficient C_l as output. The ESCNN architecture is straightforward. As shown in figure 4.1, it consists of two standard convolution layers followed by non-linear activation function, and a fully connected layer before the prediction [58]. The two convolution layers use 200 convolution filters, where the filter size of first layer is 5×1 and the second layer is 1×1 respectively. The last layer, fully connected layer, contains 159

neurons. Note that since the ESCNN takes sequential airfoil coordinates as input, and only convolutions and activations are involved in the architecture, therefore all the hidden layers are not permutation invariant.

The airfoil data samples are taken from the database of UIUC Applied Aerodynamic Group [31], which covers a wide range of airfoil types from real-world designs. Each airfoil is represented by 160 points in x, y format. For the flow conditions, the Mach number $Mach$ and Reynolds number Re are fixed to 0.3 and 1.3×10^7 respectively, and the angle of attack α varies from -2° to 10° to avoid flow separation (laminar flow). The ground truth of C_l is computed with the CFD solver Xfoil [33], for each angle of attack at the specified flow condition for each airfoil geometry. In total 15678 samples are generated to create the final dataset. The whole dataset is divided into training set and validation set at the ratio of 8:2. Note that in laminar flow where the dataset is generated, the lift coefficient C_l varies linearly with the angle of attack α , however the relationship between C_l and the airfoil coordinates is highly-nonlinear [3].

Table 4.1 shows the performance benchmarks of different models, where the regression models are provided as baselines, and three types of NNs are compared. Experiments are implemented on the Pytorch and MindSpore open-source deep learning frameworks. The Multi-layer Perceptron (MLP) network model has three hidden layers, where each layer contains 256,128,128 neurons respectively. The AeroCNN is a recent NN model that achieves state-of-the-art prediction accuracy [1]. In the AeroCNN framework, the airfoils are processed as images, such that AeroCNN can adopt the typical convolution neural network architecture for image recognition [1]. The error ε is the relative error defined by equation 4.1, where $\hat{C}_l$ denotes the predicted lift coefficient and C_l denotes the ground truth lift coefficient.

$$\varepsilon = \frac{\|\hat{C}_l - C_l\|_2}{\|C_l\|_2}. \quad (4.1)$$

It is noted that even the baseline regression models can bring the error down below 10%. However, further reducing the prediction error is a difficult task: the model's learning capacity must be large enough to accurately approximate the non-linear relationship between C_l and airfoil coordinates. Generally, the error can be further reduced at the cost of adding more parameters to increase the model's complexity [98], provided sufficient training data. However, the cost becomes more significant as the error gets smaller: the parameters are doubled to reduce the error from 5.58% (Multilayer Perceptron) to 3.46% (AeroCNN).

In contrast, the ESCNN achieves 0.97% error with two orders of parameters less than AeroCNN. Figure 4.2 shows examples of the ESCNN prediction performance on the validation

set, where the predictions are accurate and the $C_l - \alpha$ linearity is well captured. With such few parameters and limited capacity, how can ESCNN perform so well in a high-dimensional and highly-nonlinear aerodynamic system?

Table 4.1 Performance benchmarks of different models.

Model	Parameters	Error
Linear Regression	322	9.82%
MLP	961	8.43%
Multilayer Perceptron	131,969	5.58%
AeroCNN [1]	266,145	3.46%
ESCNN	1,561	0.97%

4.3 Learning the Kutta condition

While investigating how ESCNN learns to make prediction, the fully connected layer, as shown in figure 4.1, draws our attention, since it is the last hidden layer before prediction.

We track the neuron values of the fully connected layer with a test airfoil at 3° angle of attack during the training. The test airfoil NACA 2412 contains 160 coordinates. Variations of the neuron values $[f_1, f_2, \dots, f_n]$ at the end of different training stages are shown in figure 4.4, where the x-axis denote the sequential index number $[1, 2, \dots, n]$ of the neurons, and the y-axis denote corresponding neuron values $[f_1, f_2, \dots, f_n]$. Since the ESCNN model takes airfoil coordinates that are sequentially formatted as input, the order of coordinates is reflected in the neurons of learned hidden layers. The epoch numbers represent different training

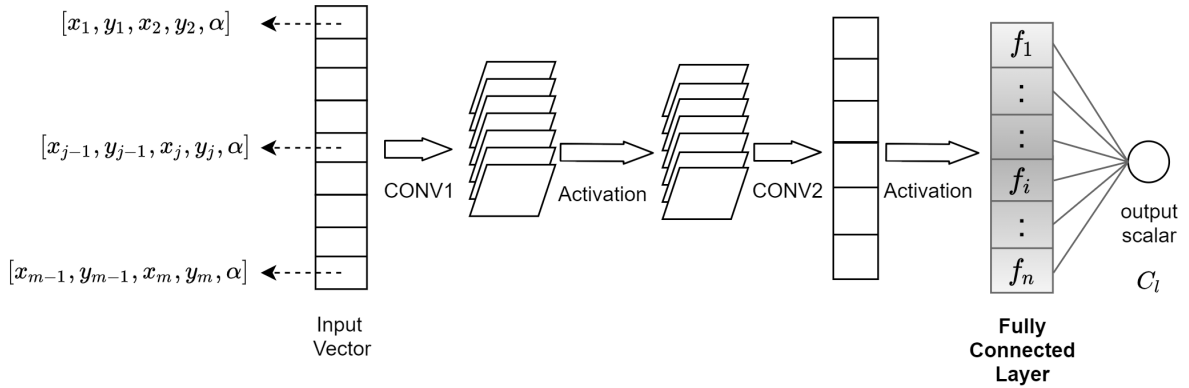


Figure 4.1 ESCNN architecture.

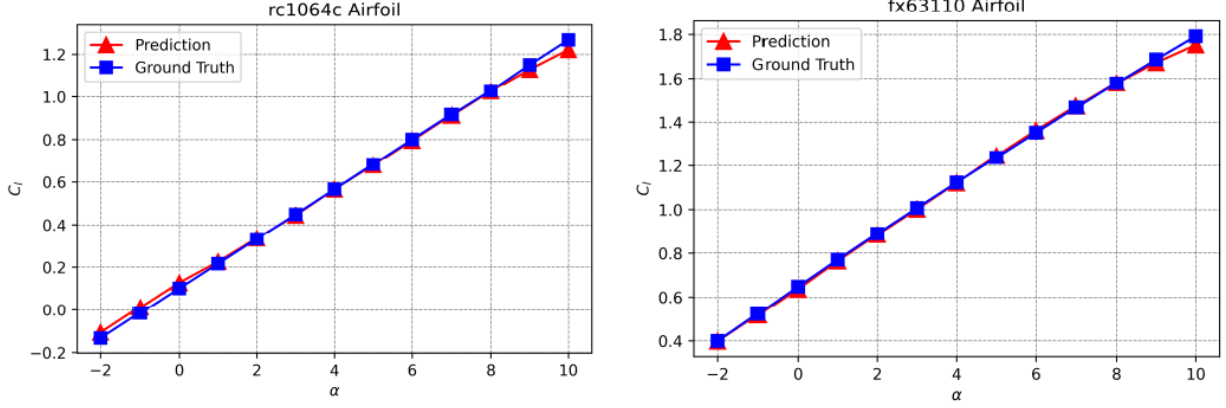


Figure 4.2 Prediction performance on validation airfoils.

iterations, the network learns and evolves with the increasing of training epoch numbers. From the figure, it is obvious that as the training progresses, the fully connected layer is converging to a sine-shaped pattern, and this pattern begins to stabilize at around 400 epochs as shown in figure 4.4. This result is also consistent with the learning curve that is shown in figure 4.3, as in this figure, after 400 epochs, the fitted Mean Squarer Error (MSE) converges towards a constant.

We notice an interesting phenomena, as shown in figure 4.4, the first and the last neurons values are always the same ($f_1 = f_n$) even at the very beginning of training. It seems like the ESCNN has learned, early in the training process, of the condition $f_1 = f_n$, and followed this rule strictly during the training progress. This $f_1 = f_n$ pattern reminds us the fundamental principle in aerodynamics: Kutta condition.

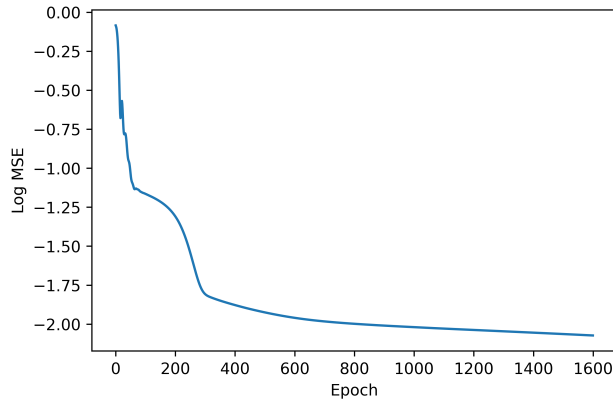


Figure 4.3 ESCNN training learning curve.

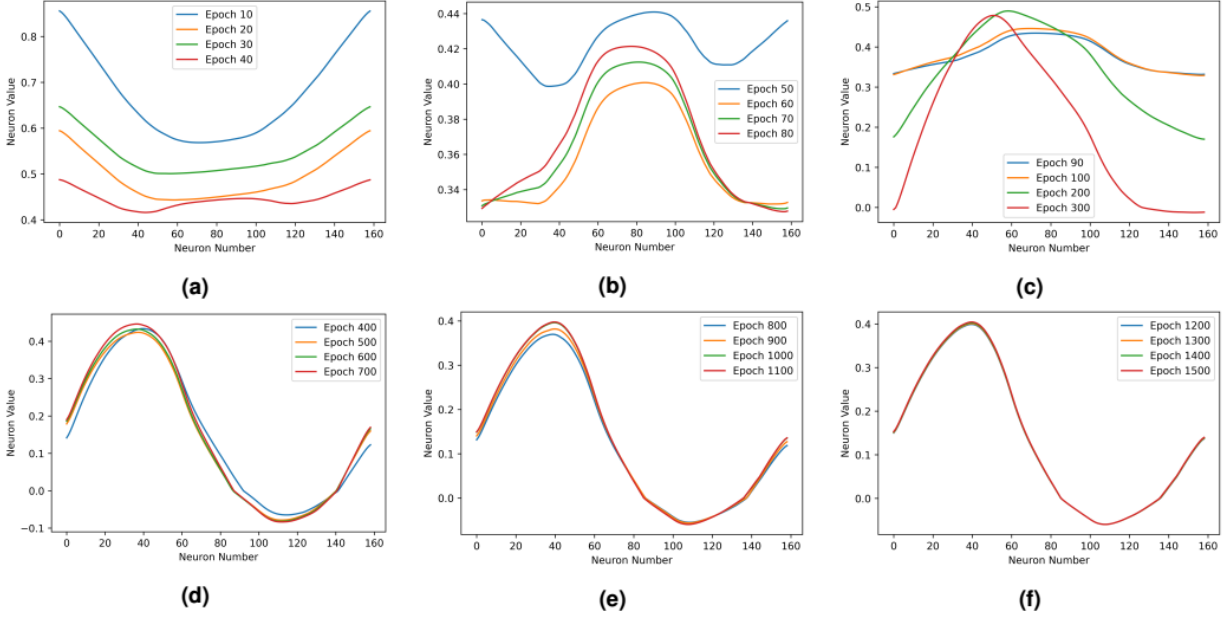


Figure 4.4 Fully connected layer evolution in training.

Recall the Kutta condition [3]: in fluid flow around a body with a sharp corner, the flow pattern in which fluid approaches the corner from both directions meets at the corner before flows smoothly go away from the body, as shown in figure 4.5. Equation (4.2) describes the Kutta condition mathematically, where γ_1 and γ_n represents the vortex strength at the upper and lower surface of the airfoil trailing edge respectively.

$$\gamma_1 = \gamma_n. \quad (4.2)$$

Apparently, ESCNN manages to figure out the importance of the Kutta condition by itself — it prioritizes to keep the value of the first neuron and the last neuron the same during the entire learning process.

4.4 Learning the vortex strength distribution pattern

To further investigate the meaning of the sine-shaped pattern, we compute the vortex strength distribution over airfoil using Vortex Panel Method (VPM). VPM is an engineering numerical method to compute the vortex strength distribution over airfoil. It replaces the airfoil surface with a series of vortex panels.

Figure 4.6 shows the scaled vortex strength distribution calculated by VPM over the test case NACA 2412 airfoil at 3° angle of attack. As shown in this figure, when fewer panels

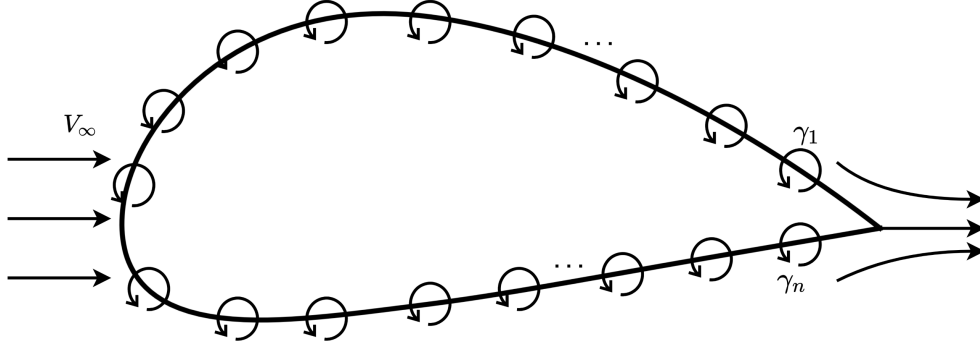


Figure 4.5 Airfoil vortex panels in free stream.

are used in the calculation, the distributions of vortex strength γ , are not smooth. The oscillations from one panel to another is a well-known flaw of VPM which is triggered by the numerical inaccuracy [3]. With the larger number of panels used during the calculation, the oscillations fades away, the results get more accurate, and the sine-shaped pattern of vortex distribution gradually emerges.

Figure 4.7 compares the computed 160 panels vortex strength distribution with the converged layer pattern at the same scale, both curves show a similar sine-shaped pattern, except they are symmetric in relation to the horizontal axis. Furthermore, if we down-sample the number of input airfoil coordinates to fewer dimensions, the converged sine-shaped pattern are still kept by the trained network, as shown in figure 4.8, not subject to the numerical inaccuracy of VPM.

Both the converged layer pattern and the computed VPM vortex strength distribution exhibit a similar sine-shaped wave. One possible hypothesis is that the network has learned another physical concept that is a function of the vortex strength. Although there is no aerodynamics law at the moment that theoretically addresses this sine-shaped pattern, we surmise that the distribution of vortex strength over airfoil follows a certain underlying pattern.

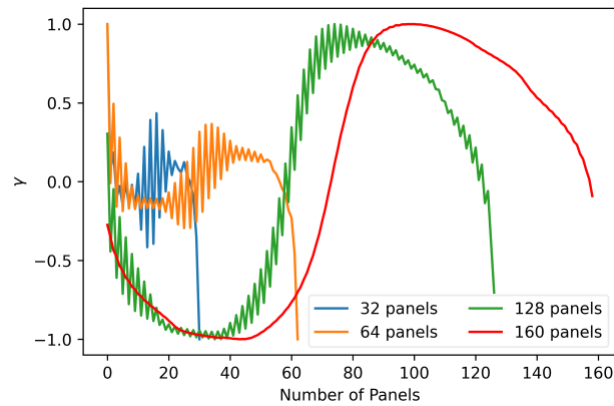


Figure 4.6 Computed vortex distribution as a function of the number of panels.

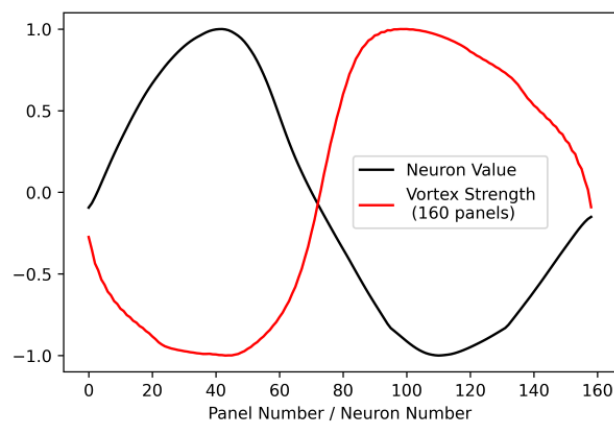


Figure 4.7 Comparison of the converged layer pattern with 160 panels vortex strength distribution.

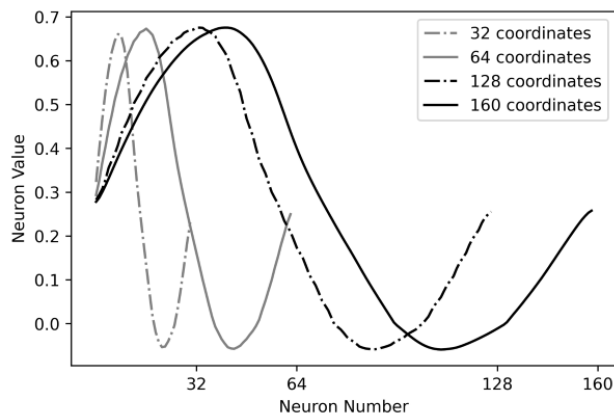


Figure 4.8 Neuron values of the fully connected layer with different number of airfoil coordinates.

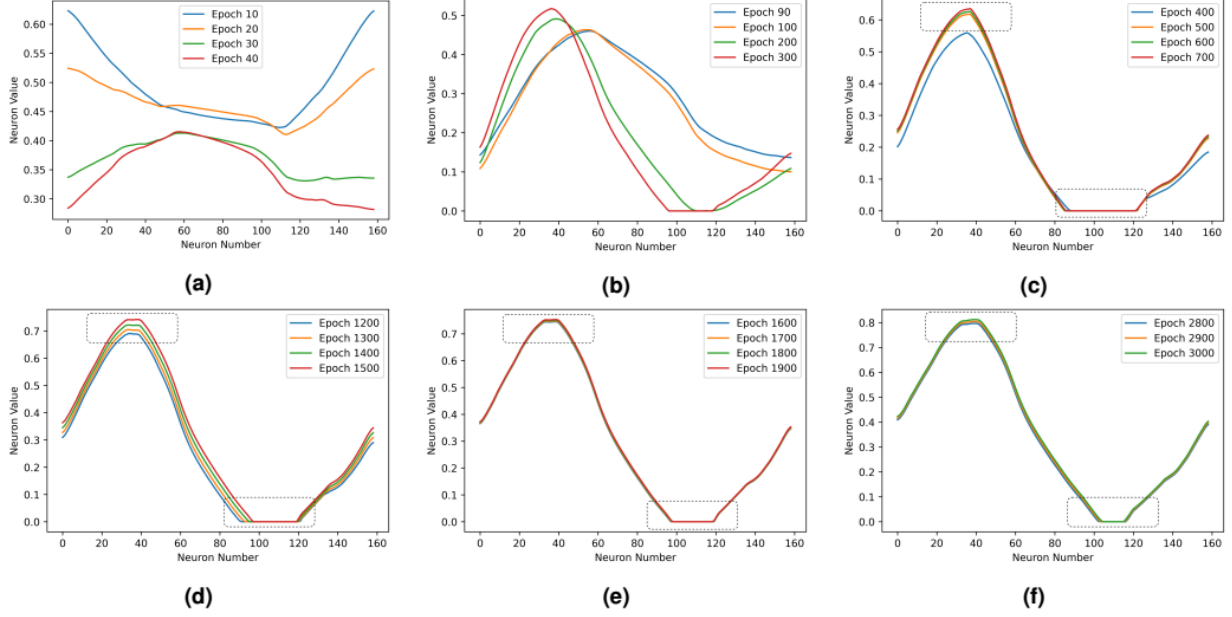


Figure 4.9 Fully connected layer under constrained activation.

4.5 Learning with ReLU activation

The initial activation function we use to train the ESCNN model is the LeakyReLU, a common choice for neural networks [99]. More importantly, the output domain of LeakyReLU activation can be both positive and negative values, so is the value of vortex strength. This feature enables ESCNN to learn the vortex related physical quantity.

However, what if the network is trained with a ReLU activation function, can it still learn the vortex related physical quantity? In this section, we intentionally limit the value range of the fully connected layer by implementing another activation — ReLU function [99]. Equation 4.3 and 4.4 describes the LeakyReLU and ReLU activation function respectively, where $\mathbf{k} = 0.5$ is the negative slope coefficient. For any real-valued input, the LeakyReLU outputs both positive and negative values, whereas ReLU outputs non-negative values only [99].

$$\text{LeakyReLU}(x) = \max(0, x) + \mathbf{k} * \min(0, x). \quad (4.3)$$

$$\text{ReLU}(x) = (x)^+ = \max(0, x). \quad (4.4)$$

Figure 4.9 shows that the ESCNN still learns the Kutta condition and vortex distribution pattern, although the pattern is constrained within the non-negative range. More interest-

ingly, the fully connected layer is forcing itself to evolve into a symmetric pattern as the training goes on, as marked in the dotted rectangular in figure 4.9. Despite that there is no constraint placed in the positive domain, the evolution to such symmetric pattern is quite clear in both figure 4.4 and figure 4.9.

4.6 Learning the lift coefficient equation

The ESCNN contains standard convolution layers, how can it outperform existing neural networks with significantly less parameters? The key reason of success is that the ESCNN makes prediction by learning the lift coefficient equation of aerodynamics at the last layer. For an airfoil at a fixed angle of attack, the lift coefficient C_l is given by equation (4.5), where c is a constant [3], γ_i is the vortex strength at the panel i , and l_i is the length of the panel i . Meanwhile, the prediction target $\hat{C}_l$ is obtained at the last layer of ESCNN (fully connected layer), as described by equation (4.6), where f_i represents the neuron values and w_i represents the learned weight parameters.

$$C_l = c \sum_{i=1}^n \gamma_i l_i. \quad (4.5)$$

$$\hat{C}_l = \sum_{i=1}^n f_i w_i. \quad (4.6)$$

We show that the ESCNN model learns equation (4.5) by matching the neuron values $[f_1, f_2, \dots, f_n]$ with the vortex strength over airfoil $[\gamma_1, \gamma_2, \dots, \gamma_n]$, and matching the learned weights $[w_1, w_2, \dots, w_n]$ with the length of each panels $[l_1, l_2, \dots, l_n]$.

We calculate the correlation coefficient between the neuron values $[f_1, f_2, \dots, f_n]$ and the vortex strength over airfoil $[\gamma_1, \gamma_2, \dots, \gamma_n]$, for each sample in the testing dataset. The correlation coefficient cov is defined by equation (4.7), where $\mathbf{x}$ and $\mathbf{y}$ are both vectors. Results show that the neuron values and vortex strength of all testing samples are highly correlated: the average of correlation coefficient is -0.975, with a standard deviation of 6.4×10^{-3} , an example is shown in figure 4.7.

$$cov = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}. \quad (4.7)$$

We also calculate the correlation coefficient between the length of each panels $[l_1, l_2, \dots, l_n]$ and the learned weights $[w_1, w_2, \dots, w_n]$. Note that the learned weights are fixed after training, however, the length of each panels $[l_1, l_2, \dots, l_n]$ varies across different airfoils. The panels

of an airfoil refer to the segments between two sequential points, as shown in figure 4.10. The panels are serially numbered from 1 to n according to their locations on the airfoil surface (starting from the trailing edge, along the upper surface to the leading edge and back around the lower surface to trailing edge). The length of each panel l_i is the spatial distance between two sequential coordinates as defined by equation 4.8. Figure 4.11 shows the randomly sampled airfoils, and their corresponding panel length distribution.

Figure 4.12 compares the learned weights and the panel length distribution of example airfoils at the same scale. Despite that panel length distributions are different across airfoils, they are close to symmetric. This symmetric pattern explains why the fully connected layer evolves for symmetry during training as shown in figure 4.4 and figure 4.9. The learned weights are not as highly correlated with the panel length distribution, as compared with the correlation between neuron values and vortex strength. However, the symmetric pattern of panel length distribution has been captured by the learned weights. Moreover, we notice from table 4.2 that the correlation between the panel length and learned weights affects the prediction performance: higher correlation leads to lower prediction error. Different airfoils have different panel length distributions, whereas the learned weights are fixed, failing to match the weights for corresponding airfoil leads to the increase of prediction error.

$$l_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}. \quad (4.8)$$

Table 4.2 Correlation (between the learned weights and the panel length) of different airfoils and prediction error.

Airfoil	Correlation	Error
GOE298	0.16	1.22%
GOE419	0.43	0.71%
EH2012	0.32	0.93%
HQ259B	0.30	0.97%
NACA632615	0.27	1.03%

How does ESCNN manages to learn the physical quantity of vortex strength and the panel length? The reason is that ESCNN adopts a structured input that incorporates prior physical knowledge.

In the vortex panel method, the vortex strength γ_1 to γ_n are obtained by solving a linear system of n equations, and solving each equation requires the angle of attack α and the coordinates of corresponding panel [3]. Inspired by the vortex panel method, we combine

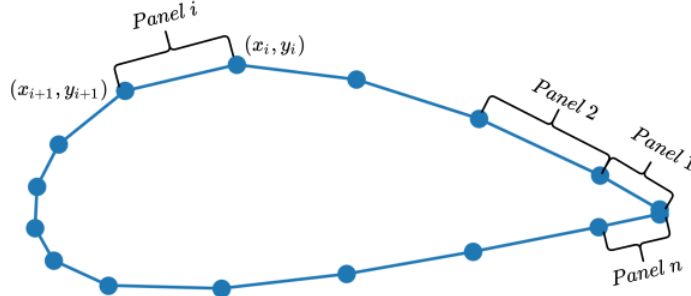


Figure 4.10 Panel distribution over the airfoil surface.

the panel coordinates $x_i, y_i, x_{i+1}, y_{i+1}$ and the angle of attack α as an element unit, and then concatenate all the element units into a single vector, as shown in figure 4.1. The first layer, CONV1, performs 1D convolution over each element unit, and each element unit contains sufficient information to solve the vortex strength and panel length.

The physics-informed structured input allows the convolution layer to pick up the vortex strength γ_i and panel length l_i , and further allows the fully connected layer to learn the lift coefficient equation.

4.7 Conclusions

In this work, we propose a neural network model that outperforms existing state-of-the-art NNs in the airfoil lift coefficient prediction task, with two orders of less parameters. We further investigate how the ESCNN makes accurate predictions.

The ESCNN network learns to constrain the first and last neurons to be equal, during the entire learning process. This is the evidence that it self-learns the fundamental aerodynamics principle, the Kutta condition. Moreover, the fully connected layer converges to a sine-shaped wave pattern that is highly correlated to the vortex strength distribution over airfoil, which demonstrates that the network learns the vortex related physical quantity. In addition, we explore ESCNN's learning ability with constrained activation by replacing LeakyReLU to ReLU function. The results show that even with a limited value range of the neurons, ESCNN can still learn the critical physics of Kutta condition and the vortex distribution pattern. In the end, we show that the network learns lift coefficient equation at the last layer.

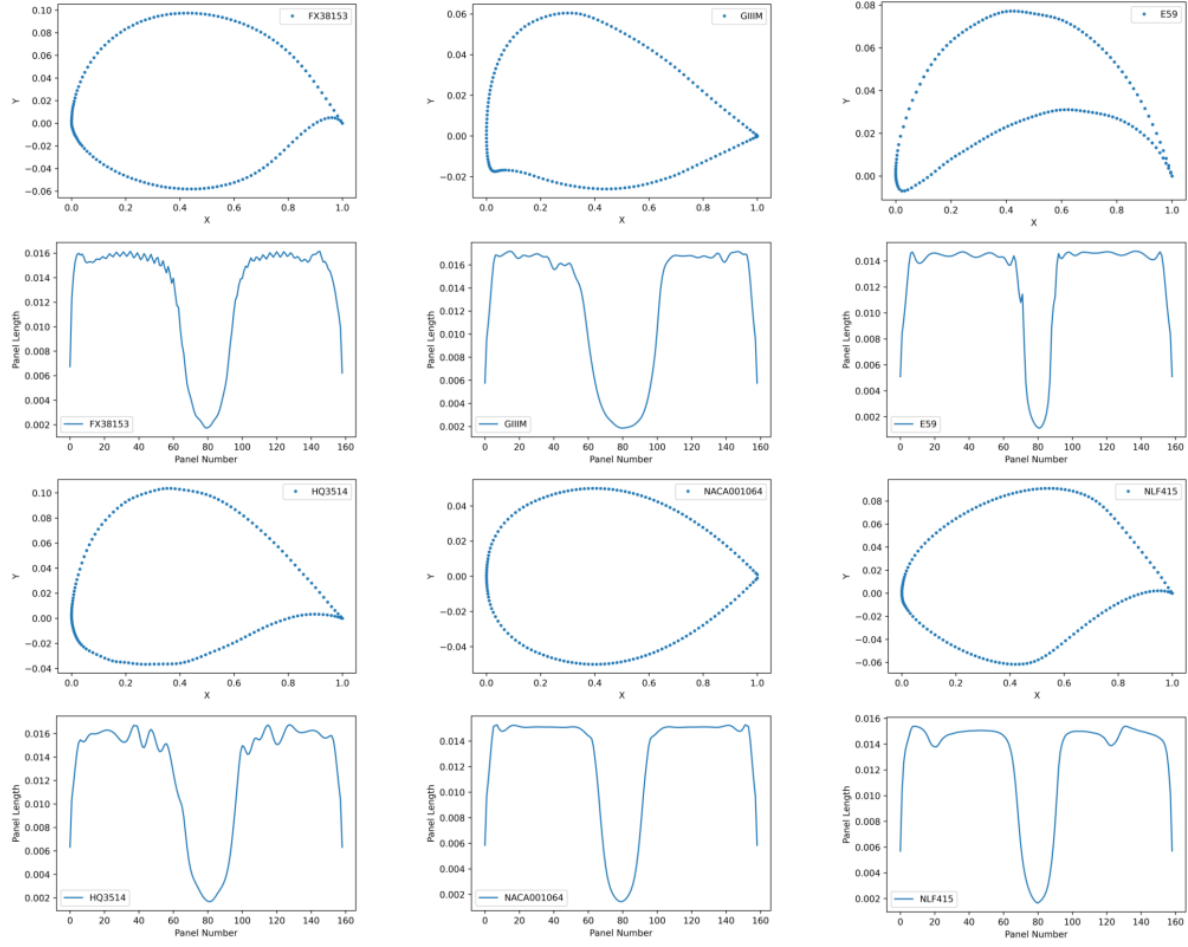


Figure 4.11 Randomly sampled airfoils and their panel length distribution.

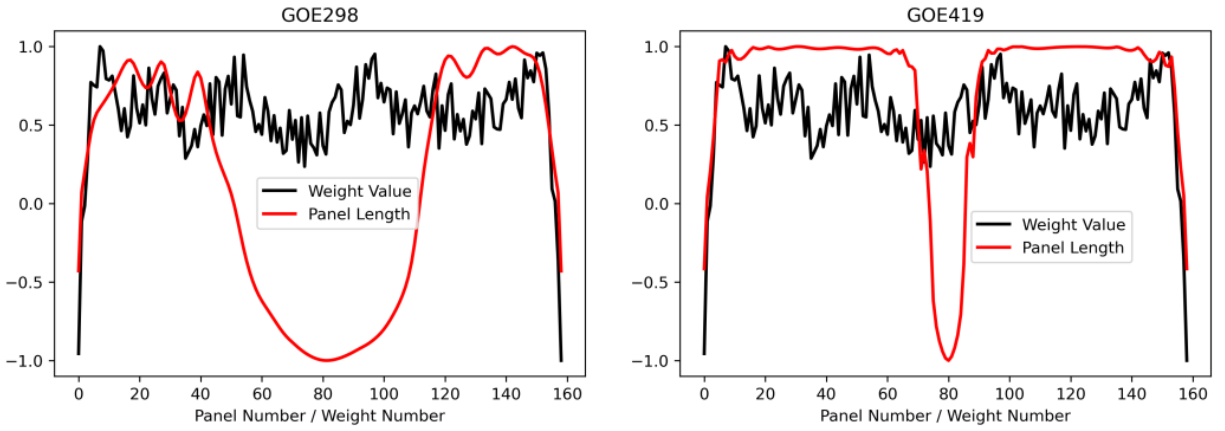


Figure 4.12 Comparison of learned weights and panel length distribution at the same scale

CHAPTER 5 ARTICLE 2: ATTENTION-ENHANCED NEURAL NETWORK MODELS FOR TURBULENCE SIMULATION

This Chapter has been published at the Physics of Fluids Journal on February 9th 2022, authored by Wenhui Peng, Zelong Yuan and Jianchun Wang.

5.1 Introduction

Over the past few years, data-driven approaches based on machine learning (ML) have been extensively explored to complement and accelerate traditional computational fluid dynamics methods [7, 8]. Most of them follows two routes: ML-assisted model-driven approaches and pure data-driven approaches. The ML-assisted model-driven approaches aim to fit the closures of classical turbulence models based on consistency with high-resolution direct numerical simulations. Ling et al. presented a method of using deep neural networks to learn a model for the Reynolds stress anisotropy tensor from high fidelity simulation data [9]. Maulik et al. used neural networks to learn the map between inputs given by stencils of the vorticity and the streamfunction along with information from the eddy-viscosity kernels, and predict the sub-grid vorticity forcing in a temporally and spatially dynamic fashion [10]. Wang et al. introduced a semi-explicit deep learning based framework for the reconstruction of the subgrid-scale stress in the large eddy simulation of incompressible turbulence [100]. Beck et al. designed networks based on local convolution filters to predict the underlying unknown non-linear mapping from the coarse grid quantities to the closure terms without prior assumptions [11]. Portwood et al. showed that artificial neural networks can provide useful insights in the process of developing and improving turbulence closures [12]. Park et al. used a fully connected neural network to develop a subgrid-scale (SGS) model in a turbulent channel flow [101]. Yuan et al. developed a dynamic iterative approximate deconvolution models with Galilean invariance for the subgrid-scale stress in the large-eddy simulation of turbulence [34].

Despite that the ML-assisted models are potentially more accurate than traditional turbulence models, they failed to achieve the desired computational expense reduction [60]. By contrast, the pure data-driven methods aim to approximate the entire Navier-Stokes equations by deep neural networks [13–16]. Once trained, the "black-box" neural network models can make inference within seconds on modern computers, thus can be extremely efficient compared with traditional approaches [17].

Jin et al. proposed a fusion convolutional neural networks using measurements of the pressure

field on the cylinder to predict the velocity field around a cylinder [62]. Sekar et al. presented a data driven approach that combines the deep convolution neural network and deep multilayer perceptron to predict the incompressible laminar steady flow field over airfoils [19]. Chen et al. proposed to apply the U-net architectures for fast prediction in laminar flows [75]. Cheng et al. used the deep residual neural network to predict the flow velocity fields [63]. Yousif et al. applied the generative adversarial network to reconstruct high-resolution turbulent flow fields [76]. Chen et al. proposed a graph neural networks architecture as a surrogate model for laminar flow prediction around two-dimensional obstacles [20]. Schmidt et al. simulated the turbulence quantities with a deep neural network to accelerate the modeling of transient flashing jets [102]. Han et al. designed a hybrid deep neural network architecture to capture the spatial-temporal features of unsteady flows from the high-dimensional numerical unsteady flow field [21]. Nakamura et al. presented a machine learning based reduced order model for three-dimensional complex flows, by combining a three-dimensional convolution neural network autoencoder and a long short-term memory network. Wang et al. developed a convolution neural network to improve the near-wall velocity field prediction [22]. Li et al. proposed a deep neural network architecture for reconstructing the slices of the two-dimensional pressure field in three-dimensional flow, where the architecture embeds the convolution into the long short-term memory network to realize the purpose of using the upstream pressure to reconstruct downstream pressure [65]. Ren et al. proposed a deep learning framework that combines the long short-term memory network and the convolution neural network for prediction of temporal evolution of turbulent flames [61].

Despite these significant achievements in the acceleration of flow simulation, these models suffer from the generalization problem, and are sensitive to the parameters change [60]. Recent progress has been made to relieve the generalization problem with prior physical knowledge [88]. Erichson et al. proposed to include the physics-informed prior knowledge to improve the model generalization performance on a fluid flow prediction task [87]. Pawar et al. developed a physics-guided machine learning framework that leverages the interpretable physics-based model with a deep learning model, to improve the generalizability of neural networks [103]. Xu et al. employed the physics-informed deep learning by treating the governing equations as a parameterized constraint to reconstruct the missing flow dynamics [104]. 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 learned representations, and achieved state-of-the-art prediction accuracy [64].

Proposed by Li et al., the Fourier neural operator (FNO) mimics the pseudo-spectral methods [105, 106]: it parameterizes the integral kernel in the Fourier space, thus directly learns the mapping from any functional parametric dependence to the solution [4]. Benefited from the expressive and efficient architecture, the FNO outperforms previous state-of-the-art neural network models: the FNO achieves 1% error rate on two-dimensional (2D) turbulence prediction task at low Reynolds numbers. However, when the fluid dynamic system becomes more chaotic, the Fourier neural operator suffers from the same problem [4]: the error rate arises over 15% at the Reynolds number of 10^5 .

Making accurate predictions of turbulence at high Reynolds numbers has always been a difficult and challenging task for neural networks. An import reason is that the existing neural network models fail to efficiently incorporate the multi-scale and nonequilibrium characteristics of the high Reynolds numbers turbulence into the design of network architecture.

We propose to model the multi-scale and chaotic properties of turbulence with the concept of visual attention. Human visual attention allows us to focus on a specific region that contains important features while perceiving the surrounding environment with less concentration. Inspired by the human visual attention, we expect the attention mechanism to capture strongly nonequilibrium regions by focusing on certain important regions in the flow field.

The attention mechanism in deep learning was first introduced by Bahdanau et al. for machine translation [107]. In recent years, attention mechanism has shown itself to be very successful in boosting the performance of neural networks on a variety of tasks, ranging from nature language processing to computer vision [29, 108, 109]. In fluid dynamics, attention mechanism has also been used to enhance the reduced order model to extract temporal feature relationships from high-fidelity numerical solutions [78]. In this work, we couple the attention mechanism with the state-of-the-art FNO model to improve the model performance on high Reynolds numbers flow.

This paper is organized as follow: section 5.2 briefly introduces the Fourier neural operator, section 5.3 shows the detailed implementation of the attention mechanism, in section 5.4 we benchmark the attention improvement with numerical experiments on the prediction task of 2D turbulence. In the sections 5.5 and 5.6 we give discussions and draw conclusions, respectively.

5.2 The Fourier neural operator

Most neural network architectures have focused on learning mappings between finite-dimensional Euclidean spaces, they are good at learning a single instance of the equation, but have dif-

difficulty to generalize well once the governing equation parameters or boundary conditions changes [88, 110–112]. Given a finite collection of the observed input-output pairs, the Fourier neural operator learns the mapping from any functional parametric dependence to the solution, meaning that they learn an entire family of partial differential equations instead of a single equation [4]. Specifically, let $D \subset \mathbb{R}^d$ be a bounded, open set, and notate the target non-linear mapping as $G^\dagger : \mathcal{A} \rightarrow \mathcal{U}$, where $\mathcal{A}(D; \mathbb{R}^{d_a})$ and $\mathcal{U}(D; \mathbb{R}^{d_u})$ are separable Banach spaces of function taking values in $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_u}$ respectively [113]. The Fourier neural operators learns an approximation of $G^\dagger$ by constructing a mapping parameterized by $\theta \in \Theta$. $G : \mathcal{A} \times \Theta \rightarrow \mathcal{U}$. The optimal parameters $\theta^\dagger \in \Theta$ are determined in the test-train setting by using a data-driven empirical approximation [114], such that $G(\cdot, \theta^\dagger) = G_{\theta^\dagger} \approx G^\dagger$.

The neural operators [17] are formulated as an iterative architecture $v_0 \mapsto v_1 \mapsto \dots \mapsto v_T$ where v_j for $j = 0, 1, \dots, T-1$ is a sequence of functions each taking values in $\mathbb{R}^{d_v}$, as shown in Fig.5.1. Firstly, the input $a \in \mathcal{A}$ is transformed to a higher dimensional representation $v_0(x) = P(a(x))$. by a fully connected layer P , then the higher dimensional representation is updated iteratively by Eq.(5.1), where $\mathcal{K} : \mathcal{A} \times \Theta_{\mathcal{K}} \rightarrow \mathcal{L}(\mathcal{U}(D; \mathbb{R}^{d_v}), \mathcal{U}(D; \mathbb{R}^{d_v}))$ maps to bounded linear operators on $\mathcal{U}(D; \mathbb{R}^{d_v})$ and is parameterized by $\phi \in \Theta_{\mathcal{K}}$, $W : \mathbb{R}^{d_v} \rightarrow \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is an elementally defined non-linear activation function. Lastly, the output $u \in \mathcal{U}$ is obtained by applying the local transformation $u(x) = Q(v_T(x))$, where $Q : \mathbb{R}^{d_v} \rightarrow \mathbb{R}^{d_u}$.

$$v_{t+1}(x) = \sigma(Wv_t(x) + (\mathcal{K}(a; \phi)v_t)(x)), \quad \forall x \in D. \quad (5.1)$$

Let $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse transform of a function $f : D \rightarrow \mathbb{R}^{d_v}$ respectively. Replacing the kernel integral operator in Eq. (5.1) by a convolution operator defined in Fourier space, and applying the convolution theorem, the Fourier integral operator can be expressed by Eq.(5.2), where R_ϕ is the Fourier transform of a periodic function $\kappa : \bar{D} \rightarrow \mathbb{R}^{d_v \times d_v}$ parameterized by $\phi \in \Theta_{\mathcal{K}}$, as shown in Fig.5.1.

$$(\mathcal{K}(\phi)v_t)(x) = \mathcal{F}^{-1}(R_\phi \cdot (\mathcal{F}v_t))(x), \quad \forall x \in D. \quad (5.2)$$

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 $k_{\max} = |Z_{k_{\max}}| = |\{k \in \mathbb{Z}^d : |k_j| \leq k_{\max,j}, \text{ for } j = 1, \dots, d\}|$. 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-

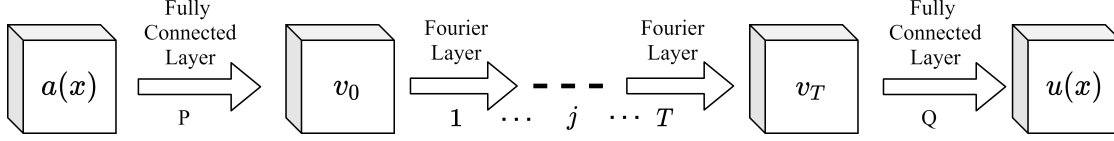


Figure 5.1 Fourier neural operator (FNO) architecture.

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 $(R_\phi \cdot (\mathcal{F}v_t))_{k,l} = \sum_{j=1}^{d_v} R_{\phi k,l,j} (\mathcal{F}v_t)_{k,j}$, $k = 1, \dots, k_{\max}$, $j = 1, \dots, d_v$.

In CFD modeling, the flow is typically uniformly discretized with resolution $s_1 \times \dots \times s_d = n$, and $\mathcal{F}$ can be replaced by the Fast Fourier Transform (FFT). For $f \in \mathbb{R}^{n \times d_v}$, $k = (k_1, \dots, k_d) \in \mathbb{Z}_{s_1} \times \dots \times \mathbb{Z}_{s_d}$, and $x = (x_1, \dots, x_d) \in D$ the FFT $\hat{\mathcal{F}}$ and its inverse $\hat{\mathcal{F}}^{-1}$ are given by Eq. (5.3), for $l = 1, \dots, d_v$.

$$\begin{aligned}
 (\hat{\mathcal{F}}f)_l(k) &= \sum_{x_1=0}^{s_1-1} \dots \sum_{x_d=0}^{s_d-1} f_l(x_1, \dots, x_d) e^{-2i\pi \sum_{j=1}^d \frac{x_j k_j}{s_j}}. \\
 (\hat{\mathcal{F}}^{-1}f)_l(x) &= \sum_{k_1=0}^{s_1-1} \dots \sum_{k_d=0}^{s_d-1} f_l(k_1, \dots, k_d) e^{2i\pi \sum_{j=1}^d \frac{x_j k_j}{s_j}}.
 \end{aligned} \tag{5.3}$$

5.3 Attention-enhanced neural network

Human visual attention allows us to focus on a specific region that contains important features while perceiving the surrounding environment with less concentration. Inspired by the concept of visual attention, we propose to model the nonequilibrium features of turbulence with the self-attention module in machine vision [2].

The idea of attention mechanism was first popularly applied in the domain of natural language processing [29]. 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 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 [29]. The three sub-modules (query, key, and value) are the pivotal components of attention mechanism, which come from the concepts of information retrieval systems [115]. An example is the query search, where the search engine maps the query to a set of keys, and output the values. Zhang et al. extended the idea into the domain of computer vision by proposing the self-attention module [2]. The overall architecture of the self-attention block is shown in Fig.5.2, it learns sequential attention maps that are called the channel attention and the space attention. The

first layer performs 1×1 convolutions on $v_t(\mathbf{x})$ to sequentially infer the channel attention map $\mathbf{Mc}$ and the space attention map $\mathbf{Ms}$, as shown in Eq.5.4. The self-attention operation mimics the “proportional retrieval” approach from information retrieval systems, where s_{ij} corresponds to the probability vector [115].

$$\mathbf{Mc} = W_h v_t(\mathbf{x}), \mathbf{Ms} = \frac{\exp(s_{ij})}{\sum_{i=1}^N \exp(s_{ij})}, \text{ where } s_{ij} = (W_f v_t(\mathbf{x}_i))^T (W_g v_t(\mathbf{x}_j)). \quad (5.4)$$

The convolution parameters W_f, W_g, W_h learn the embedding of query, key and value, respectively. Eq. 5.5 gives the attention refined feature maps, where $\otimes$ denotes the dot product.

$$v_t(\mathbf{x})' = \mathbf{Ms} \otimes \mathbf{Mc}. \quad (5.5)$$

The sequence of functions $v_0 \mapsto v_1 \mapsto \dots \mapsto v_T$ is the core structure of Fourier network, which directly affects the model learning capability. In the original network, $v_t \mapsto v_{t+1}$ is updated by Eq. 5.1, and we use the self-attention block to refine the FNO output as feature augmentation as shown in Fig.5.3. The self-attention block takes the feature maps of Fourier neural operators $v_t(\mathbf{x})$ as input, and output the attention refined feature maps $v_t(\mathbf{x})'$.

Despite that the self-attention module allows feature refinement for any neural operators $v_t \Rightarrow v_t'$ for $t = 1, \dots, T$, our numerical experiments show that a single block at the end of sequence $v_T \Rightarrow v_T'$ has the same level of improvement as refining all neural operators in the sequence $v_1 \Rightarrow v_1' \mapsto \dots \mapsto v_T \Rightarrow v_T'$.

The attention parameters W_f, W_g, W_h can be jointly learned with the Fourier layer during training. More importantly, since the attention block adopts 1×1 convolution filters, it retains the mesh invariant property of neural operators, meaning that the attention-enhanced FNO still keeps the ability of training on coarse grid and making inference on finer grid without seeing any higher resolution data. Details are discussed in Sec. 5.4.4.

5.4 Performance Benchmark

In this section, we benchmark the performance of attention-augmented neural network against the original FNO model with various criteria, including the vorticity prediction error, the vorticity spectrum, the probability density function (PDF) for vorticity and the condition average error of the vorticity. We conduct numerical experiments to evaluate the prediction ability of the two models (FNO, FNO+Attention) on the same dataset of 2D incompressible

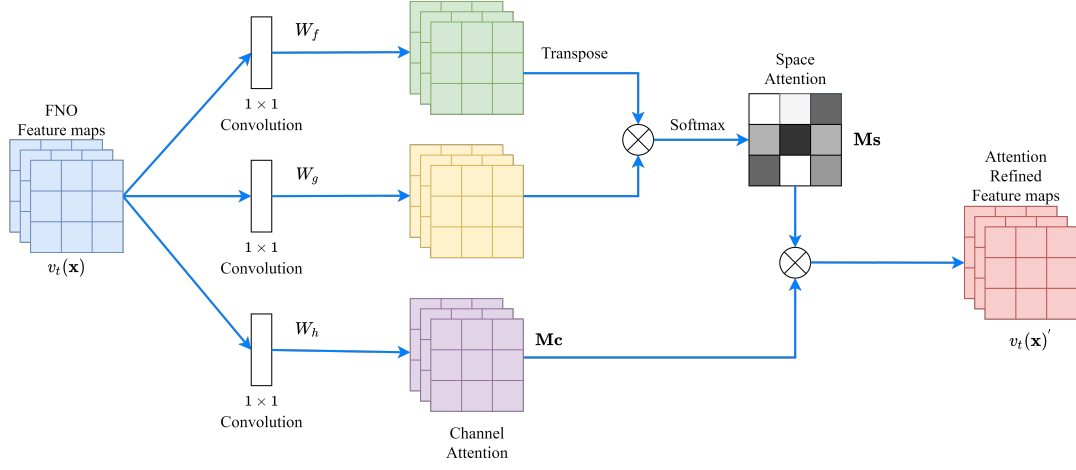


Figure 5.2 Self-attention block.

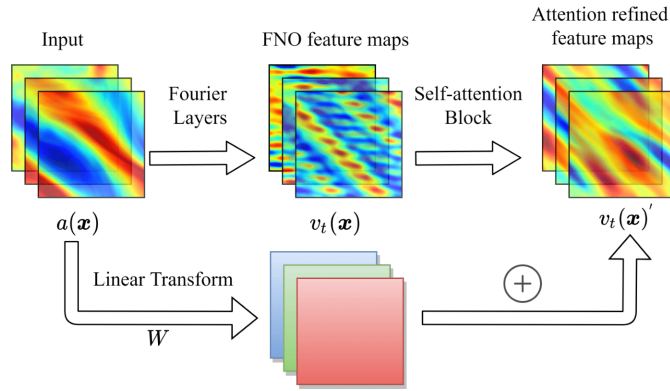


Figure 5.3 Attention enhanced Fourier neural operator (FNO).

turbulence with reference [4]. We also compare the performance of two models on more complex turbulent flows: the Taylor-Green turbulence and free shear turbulence. Specifically, we are interested in comparing the prediction accuracy of the two models at different time steps and at different Reynolds numbers.

5.4.1 Dataset description

Eq. (5.6) describes the dimensionless Navier-Stokes equations for a viscous, incompressible fluid on the unit torus, where $u \in C([0, T]; H_{\text{per}}^r((0, 1)^2; \mathbb{R}^2))$ for any $r > 0$ is the velocity field, $\omega = \nabla \times \mathbf{u}$ is the vorticity, $\omega_0 \in L_{\text{per}}^2((0, 1)^2; \mathbb{R})$ is the initial vorticity, Re is the Reynolds number, and $f \in L_{\text{per}}^2((0, 1)^2; \mathbb{R})$ is the forcing function.

$$\begin{aligned} \partial_t \omega(\mathbf{x}, t) + \mathbf{u}(\mathbf{x}, t) \cdot \nabla \omega(\mathbf{x}, t) &= (1/Re) \Delta \omega(\mathbf{x}, t) + f(\mathbf{x}), \\ \omega(\mathbf{x}, t) &= \omega_0(\mathbf{x}), \\ \mathbf{x} \in (0, 1)^2, t \in [0, T]. \end{aligned} \quad (5.6)$$

The initial condition $\omega_0(\mathbf{x})$ is generated according to $\omega_0 \sim \mu$ with periodic boundary conditions [4], where the Gaussian random distribution $\mu = \mathcal{N}(0, 7^{3/2}(49 - \Delta)^{-2.5})$ is expressed as

$$\mu(\mathbf{x}) = \frac{1}{2\pi \times 7^3(49 - \Delta)^{-5}} \exp \left[-\frac{x_1^2 + x_2^2}{2 \times 7^3(49 - \Delta)^{-5}} \right]. \quad (5.7)$$

Here, $\Delta = 2\pi/N$ represents the uniform grid spacing.

Fig. 5.4 shows the initial vorticity of ten random conditions, which are sampled from the training set and the testing set. The forcing is kept fixed $f(\mathbf{x}) = 0.1 [\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2))]$.

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

$$\omega(\mathbf{x}, t) = \sum_{\mathbf{k}} \hat{\omega}(\mathbf{k}, t) \exp(i\mathbf{k} \cdot \mathbf{x}), \quad (5.8)$$

where i stands for the imaginary unit, namely, $i^2 = -1$, $\mathbf{k} = [k_x, k_y]^T$ is the wavenumber vector, and the superscript “ $\bullet$ ” represents the variable in Fourier space. The vorticity equation in wavenumber space can be derived as

$$\left(\frac{d}{dt} + \frac{1}{Re} k^2 \right) \hat{\omega}(\mathbf{k}, t) = \hat{f}(\mathbf{k}, t) - \mathbf{u}(\mathbf{x}, t) \widehat{\bullet \nabla \omega}(\mathbf{x}, t), \quad (5.9)$$

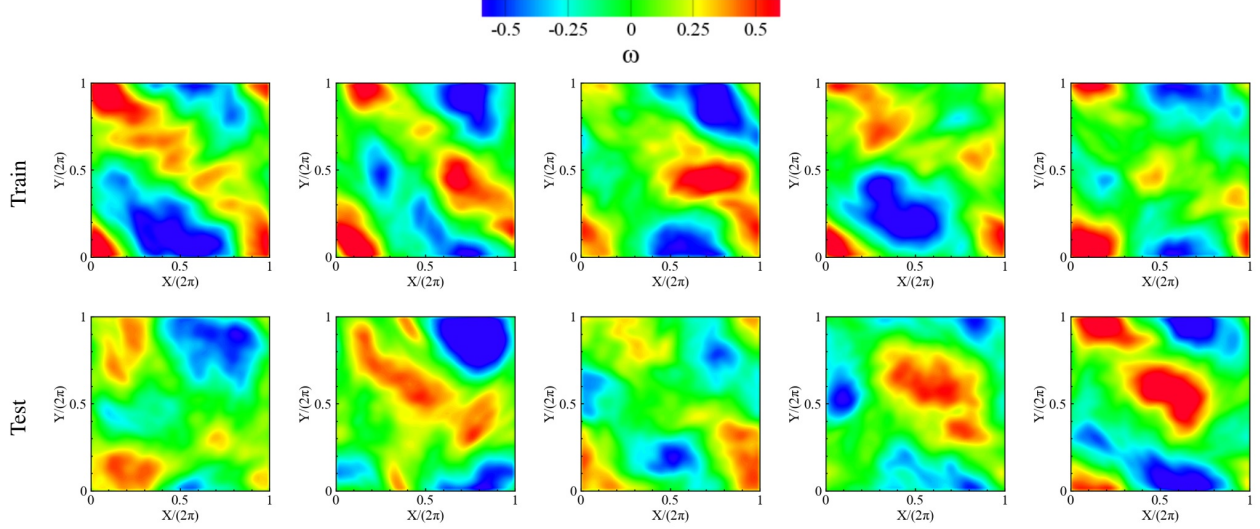


Figure 5.4 Random sampled initial conditions from the training set and testing set.

The nonlinear advection term is calculated by the pseudo-spectral method [4]. The basic procedure is to first transform the Fourier variables $\hat{\mathbf{u}}(\mathbf{k}, t) = [-ik_y \hat{\omega}/k^2, ik_x \hat{\omega}/k^2]^T$ and $\hat{\omega}(\mathbf{k}, t)$ to $\mathbf{u}(\mathbf{x}, t)$ and $\nabla \omega(\mathbf{x}, t)$ in physical space by the inverse fast Fourier transform, and perform the multiplication in physical space to obtain $\mathbf{u}(\mathbf{x}, t) \bullet \nabla \omega(\mathbf{x}, t)$, after that the forward fast Fourier transform is calculated and aliasing error is removed by the 3/2 zero-padding rule [4], respectively. Data are generated on the grid size of 256×256 and are downsampled to 64×64 . Time is advanced with the Crank-Nicolson scheme, where the time-step is set to be 10^{-4} , and the solution is recorded every $t = 1$ time units¹. For a partial differential equation $\partial_t \hat{\omega} = R(\hat{\omega}, t)$, the iterative scheme for time advancement is given by Eq. 5.10, where Δt is the time step, $t_n = n\Delta t$, and $\hat{\omega}^n$ is the vorticity at time t_n .

$$\hat{\omega}^{n+1} = \hat{\omega}^n + \Delta t \left[\frac{3}{2} R(\hat{\omega}^n, t_n) - \frac{1}{2} R(\hat{\omega}^{n-1}, t_{n-1}) \right]. \quad (5.10)$$

5.4.2 Performance benchmark at different time steps

Since the model prediction errors are produced and accumulated at every step, prediction error increases dramatically with time due to the chaotic features of turbulence. Reducing the accumulated prediction errors on time dimension is therefore still one of the most challenging problems for surrogate models.

In this numerical experiment, we generate 1200 pairs of input-output data with the numerical

¹The Adams-Bashforth scheme in the original publication is a typo and has been removed

solver, where each sample contains 20 steps of solutions of a random initialized condition at Reynolds number 10^5 . The solution is recorded every $t = 1$ time units. Both models (FNO, FNO+Attention) take the vorticity at previous 10 time steps solutions as input, and gives the vorticity at the next time step as output. During training, the vorticity of first 10 steps $\omega|_{(0,1)^2 \times [0,10]}$ 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 next 10 steps $\omega|_{(0,1)^2 \times [11,20]}$, which are labeled as ground truth.

It is worth noting that the predicted vorticity at each step is recurrently treated as ground truth and reused as the inputs with the advance of time, thus the prediction errors are accumulated iteratively. We use 1000 samples for training and 200 samples for testing. After training, we evaluate both models on the test dataset, and compare their performance at three selected time steps ($t=11$, $t=15$, $t=20$). Fig. 5.5 compares the predicted vorticity and the absolute 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 at regions where the vorticity has dramatically changes. In contrast, the errors of attention-enhanced FNO are visibly smaller in terms of the region and magnitude. We find the same phenomenon in Fig.5.9 as we investigate the relative error ϵ of each time step. The relative error ϵ is defined by Eq. 5.11, where $\tilde{\omega}$ is the predicted vorticity and ω is the ground truth vorticity.

$$\epsilon = \frac{\|\tilde{\omega} - \omega\|_2}{\|\omega\|_2}, \text{ where } \|x\|_2 = 1/n \sqrt{\sum_{k=1}^n |x_k|^2}. \quad (5.11)$$

The ensemble-averaged vorticity spectrum using 200 test samples $\overline{E_\omega(k)}$ is also compared in Fig.5.6: the predicted vorticity spectrum of both models can agree well with the ground truth in the low-wave number region. However, the FNO predicted spectrum deviates significantly from the ground truth at high-wave number region as time advances. In contrast, the attention enhanced model can accurately capture the small-scale flow structures and well reconstruct the vorticity spectrum at different flow scales. Fig.5.7 illustrates the PDFs of the normalized predicted vorticity on different time steps, where the vorticity is normalized by the root-mean-square values of ground truth vorticity. The predictions of both models have a good agreement with the ground truth in the beginning. However, as time advances, the predicted PDFs of both models become narrower, with the attention-enhanced FNO being closer to the ground truth.

Fig.5.8 shows the average of normalized absolute error conditioned on normalized vorticity at different time steps, where both vorticity and absolute error are normalized by the root-

mean-square values of ground truth vorticity. We notice that the prediction errors of both models increase with the advance of time, and the attention-enhanced FNO achieves better performance with smaller error in the whole vorticity range. As can be seen from Fig.5.9, where the spatial-averaged relative errors are plotted, the attention-enhanced model has achieved over 40% error reduction throughout all time steps.

5.4.3 Performance benchmark at different Reynolds numbers

Since the multi-scale flow structures become more chaotic and turbulent with the increasing of Reynolds numbers, reconstruction of turbulence at high Reynolds numbers has always been a challenging task for neural networks.

Table 5.1 shows the prediction relative errors (defined in Eq.5.11) of existing state-of-the-art neural network models including the FNO [4]. Note that the results of U-Net [116], TF-Net [64] and Resnet [117] are cited from reference [4]. It is noted that all existing models perform well at low Reynolds number ($Re = 10^3$), and fails at higher Reynolds numbers ($Re = 10^4$ and $Re = 10^5$), with FNO achieving the lowest error. In contrast, the attention-enhanced FNO model further reduces the prediction errors of FNO by 40% at higher Reynolds numbers.

Here, we benchmark the prediction performance of FNO and attention-enhanced FNO, at the same time step ($t=15$), for different Reynolds numbers. We train and test both models on three groups of data where the flow Reynolds number is set to $10^3, 10^4, 10^5$ respectively.

Fig.5.10 compares the snapshots of predicted vorticity and the absolute errors of the vorticity at different Reynolds numbers. Both models can accurately reconstruct the instantaneous spatial structures of turbulence at small Reynolds number $Re = 10^3$. However, when the flow becomes more turbulent at higher Reynolds numbers $Re = 10^4$ and 10^5 , the performance improvement of attention-enhanced FNO becomes more significant.

Table 5.1 Prediction errors of neural network models at different Reynolds numbers

Model	Parameters	$Re = 10^3$	$Re = 10^4$	$Re = 10^5$
FNO + Attention	466,222	0.0096	0.0855	0.0864
FNO [4]	465,717	0.0094	0.1479	0.1507
U-Net [75]	24,950,491	0.0245	0.2051	0.1982
TF-Net [64]	7,451,724	0.0225	0.2253	0.2268
ResNet [117]	266,641	0.0701	0.2871	0.2753

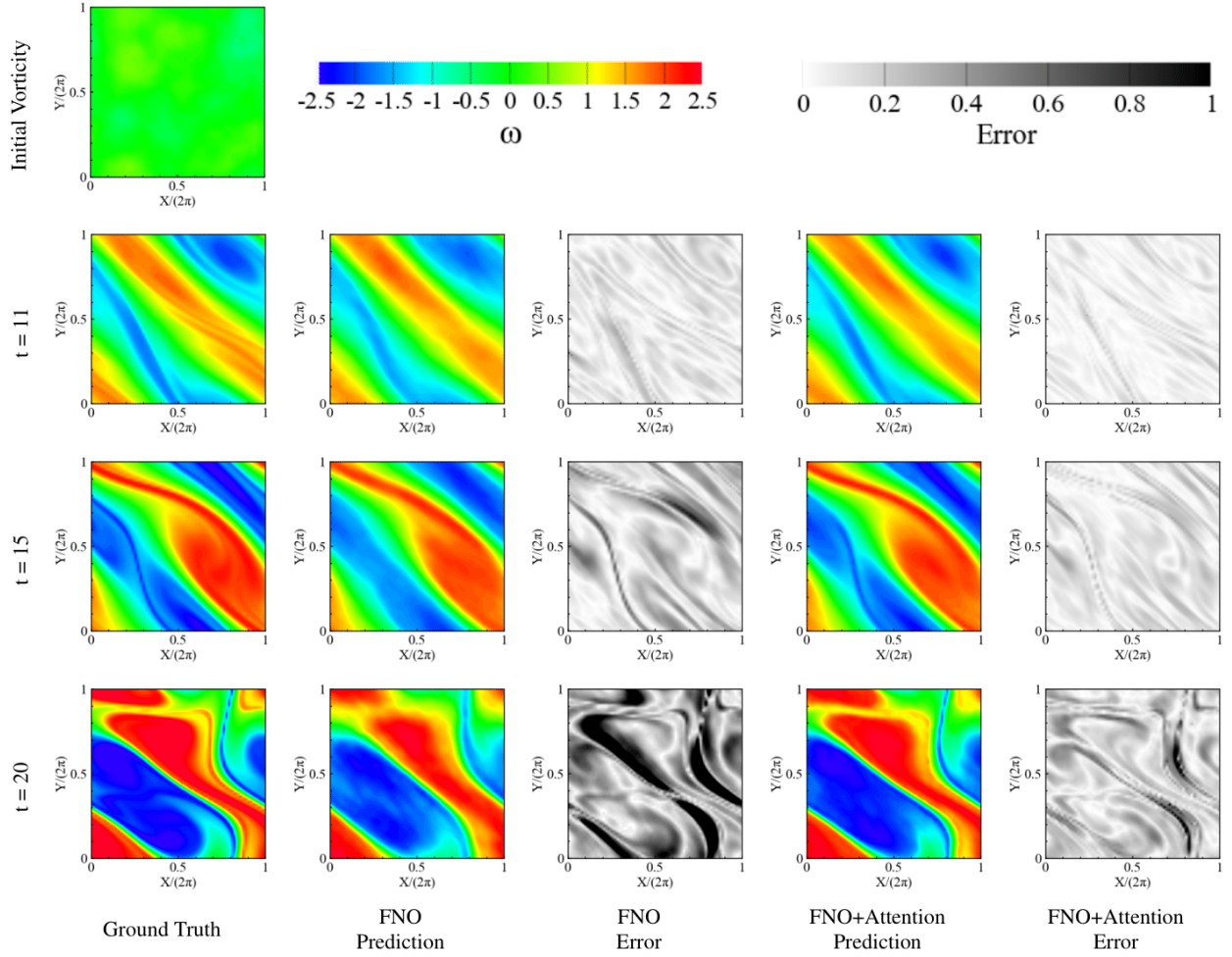


Figure 5.5 Vorticity prediction and absolute error at selected time steps.

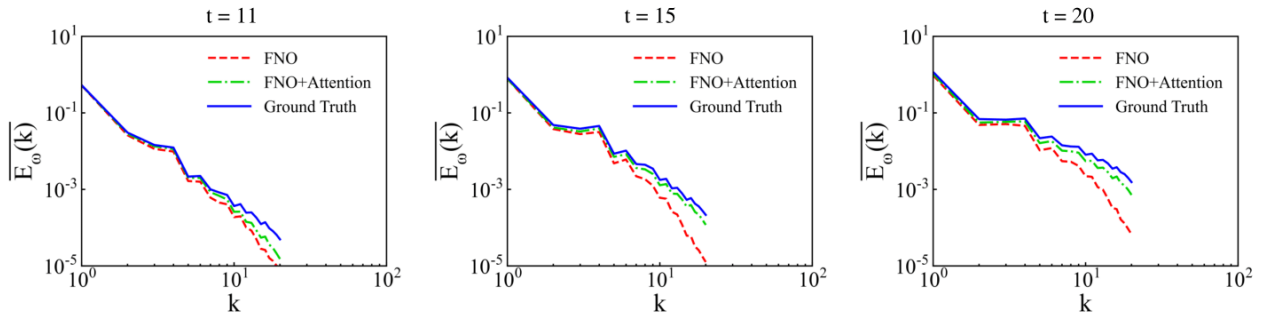


Figure 5.6 Averaged vorticity spectrum on 200 test samples.

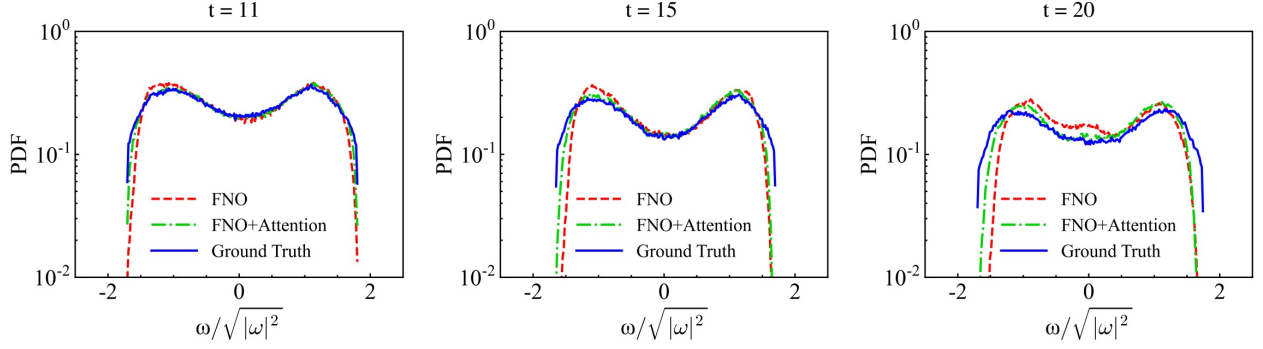


Figure 5.7 PDF of normalized vorticity at different time steps.

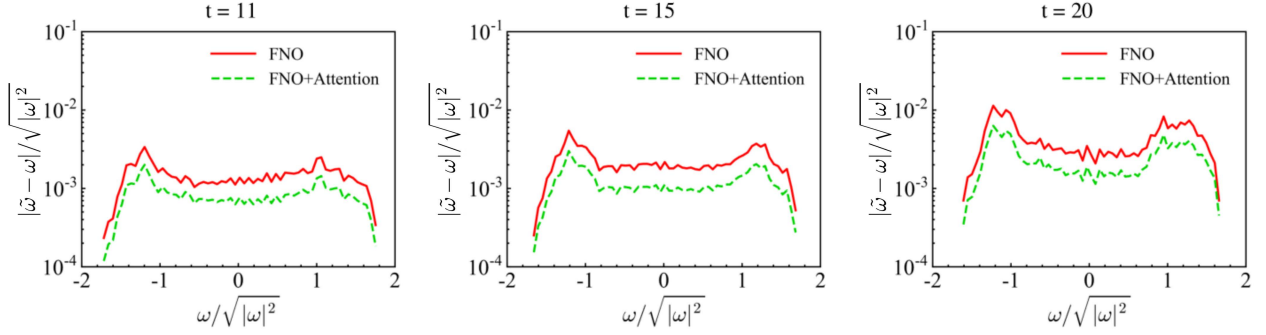
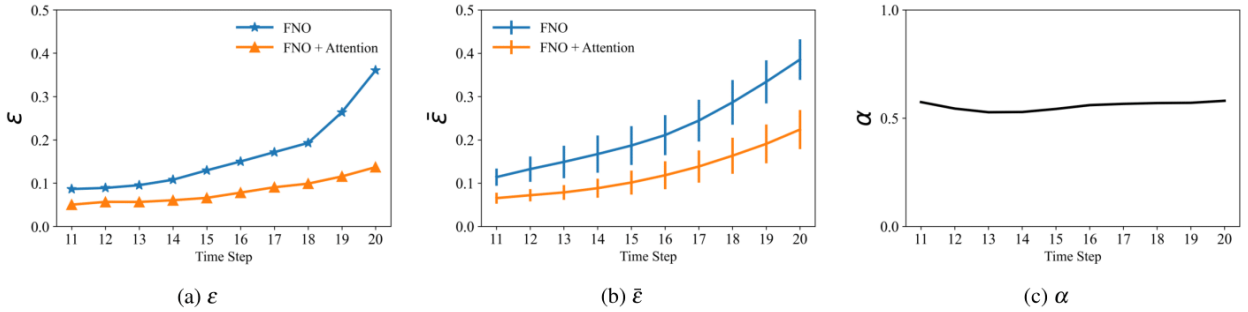


Figure 5.8 Condition average error of vorticity at different time steps.



(a) Spatial-averaged relative error of vorticity on single test sample. (b) Spatial-averaged relative error of vorticity on 200 test samples. (c) Ratio of mean relative error.

Figure 5.9 Relative error comparison at consequent time steps.

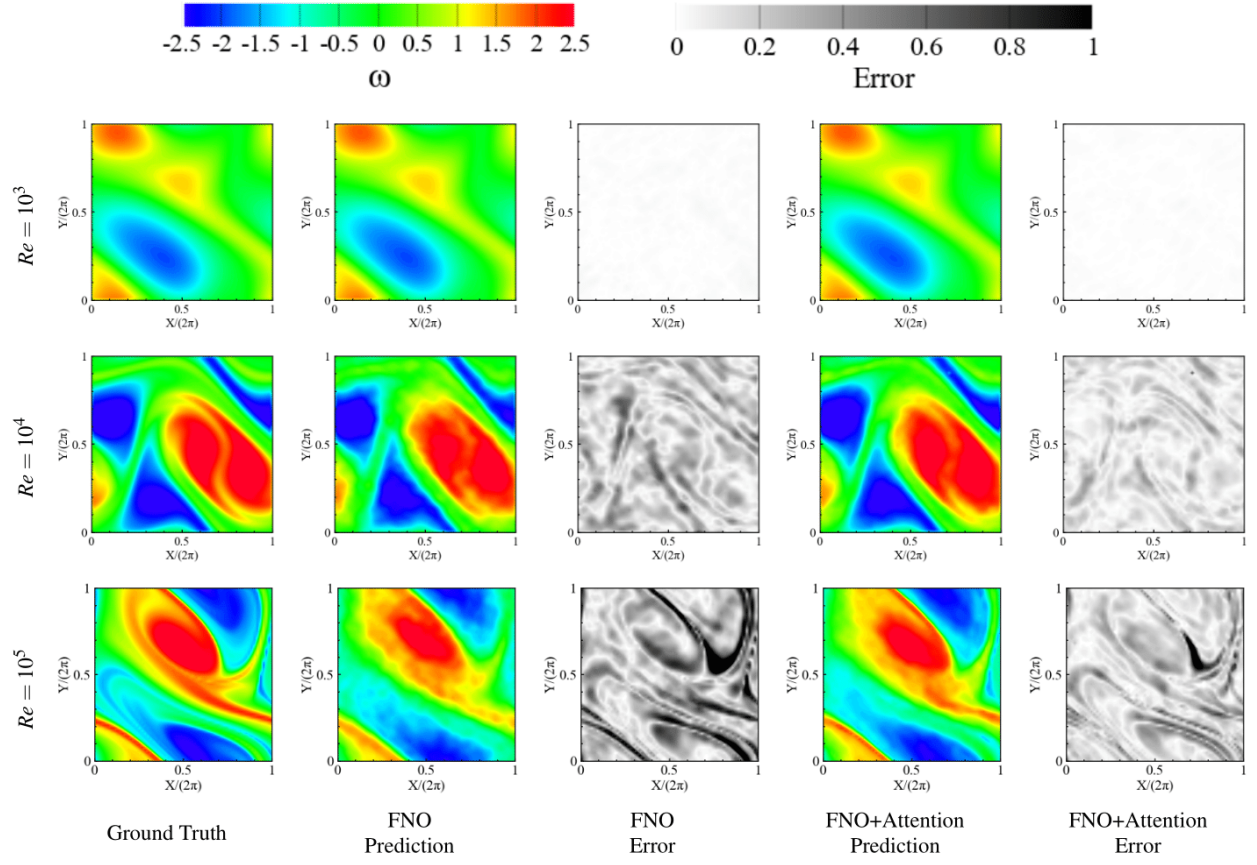


Figure 5.10 Vorticity prediction and absolute error at different Reynolds numbers.

Fig.5.11 compares the ensemble-averaged vorticity spectrum of 200 test samples at different Reynolds numbers. At small Reynolds number $Re = 10^3$, both models can reconstruct the multi-scale flow statistics accurately. However, the vorticity spectrum predicted by FNO deviates away from the ground truth as the Reynolds number increases to $Re = 10^4$ and 10^5 . In comparison, the attention-enhanced FNO can accurately capture the flow statistics at different scales.

Fig.5.12 illustrates the PDFs of normalized predicted vorticity at different Reynolds numbers. At small Reynolds number ($Re = 10^3$), the predictions of both models have good agreements with the ground truth PDF. As the Reynolds number increases to $Re = 10^4$, the predicted PDFs of both models get narrower than the ground truth; and the gap is further enlarged as the Reynolds number increases to $Re = 10^5$.

Fig.5.13 shows the conditional average of normalized absolute error at different Reynolds numbers. The prediction errors of both models become larger as the Reynolds number increase from $Re = 10^3$ to $Re = 10^4$ and $Re = 10^5$, and the performance improvement of attention-enhanced FNO also becomes more obvious with the increasing of Reynolds numbers.

Fig.5.14 compares the mean and standard deviation of relative error on 200 test samples. Both models can achieve accurate predictions (1% error) at small Reynolds number $Re = 10^3$; however, as the Reynolds number increases from 10^3 to 10^4 , the FNO prediction error arises from 1% to around 15%, whereas the FNO+Attention prediction error is about 8%; when the Reynolds number increases from 10^4 to 10^5 , the mean errors of both models are nearly unchanged except that the error standard deviation becomes larger.

In addition to benchmark the approximation capacity of the two models, we also benchmark the model generalization ability across different Reynolds numbers. We train both models on $Re = 10^5$ dataset, and test the trained models on unseen Reynolds numbers ranging from

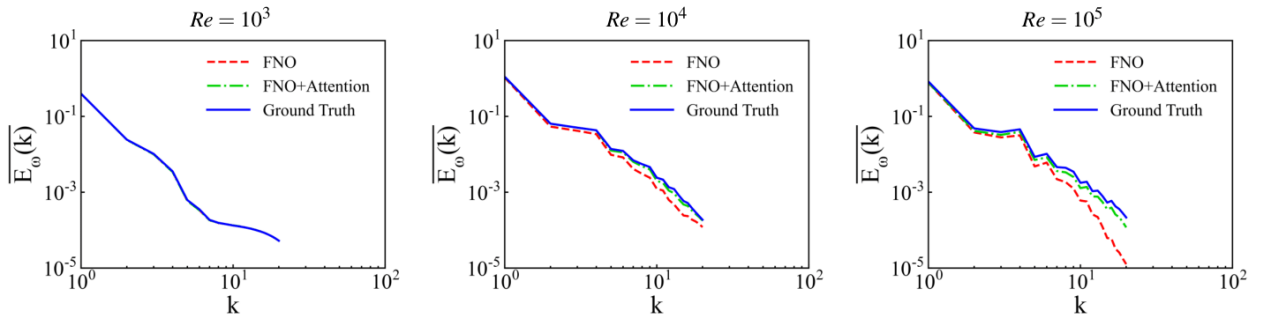


Figure 5.11 Average spectrum at different Reynolds numbers.

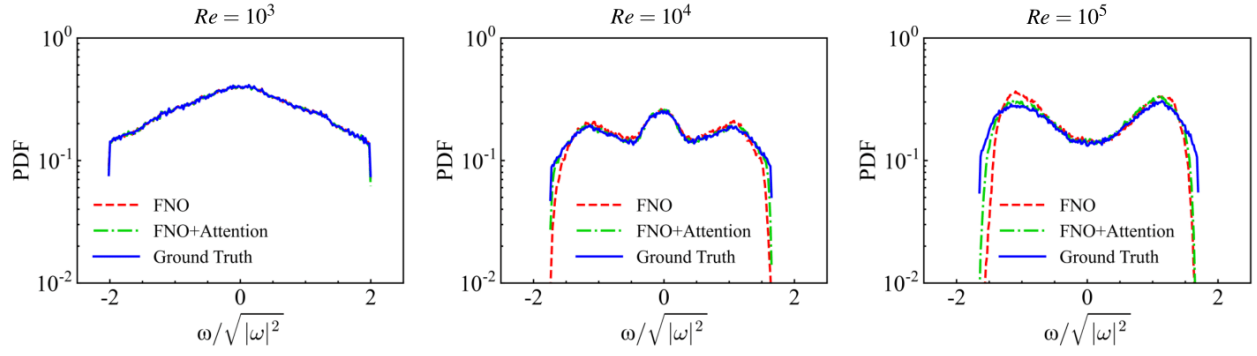


Figure 5.12 PDF of normalized vorticity at different Reynolds numbers.

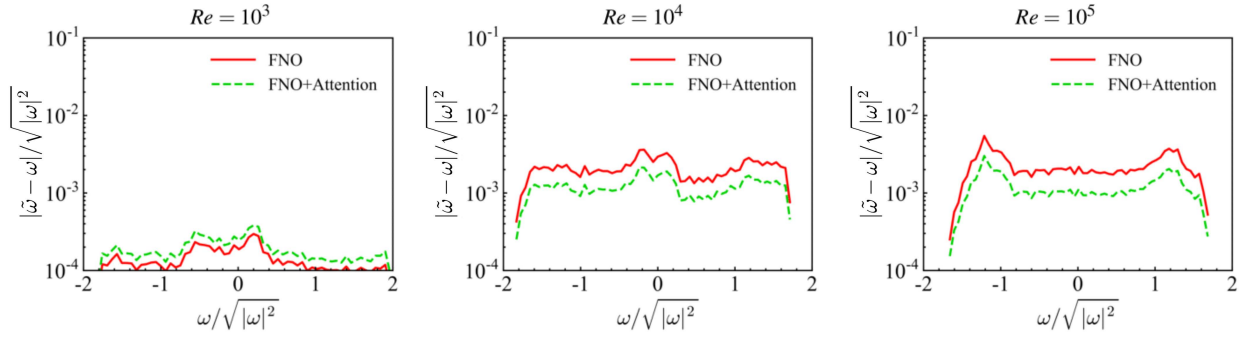


Figure 5.13 Condition average error of vorticity at different Reynolds numbers.

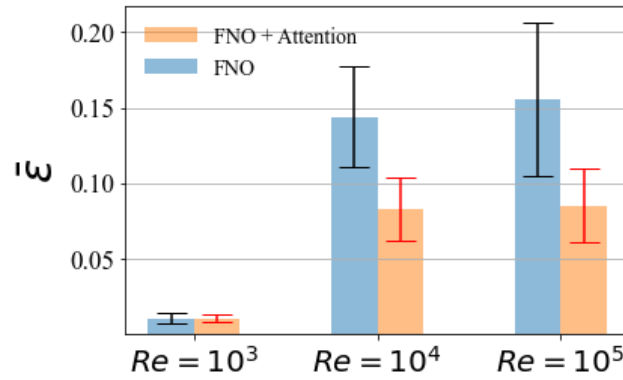


Figure 5.14 Average relative error of vorticity at different Reynolds numbers.

$Re = 10^3$ to $Re = 10^6$.

Fig.5.15 shows the generalization errors of FNO and attention-enhanced FNO models: both models can generalize well at high Reynolds numbers, whereas FNO generalizes better at low Reynolds numbers (below $Re = 5 \times 10^3$), and attention-enhanced FNO generalizes better at high Reynolds numbers. This result shows that the attention mechanism can capture the multi-scale characteristics of the high Reynolds numbers turbulence better as we expected.

Fig.5.16 shows the vorticity prediction and absolute error where the models are trained on $Re = 1 \times 10^5$ dataset and tested on three samples with different Reynolds numbers at $t = 15$. The test samples are initialized with the same random state, where the corresponding Reynolds numbers are set to $Re = 10^3$, $Re = 10^4$, and $Re = 10^6$ respectively. Both models fail to reconstruct the flow structures at low Reynolds number $Re = 10^3$, with FNO having smaller error than attention-enhanced FNO. By contrast, when the models are tested at high Reynolds numbers $Re = 10^4$ and $Re = 10^6$, the attention-enhanced FNO produces smaller generalization error than FNO, and can accurately reconstruct the multi-scale flow structures of turbulence.

5.4.4 Zero-shot super-resolution benchmark

The Fourier layers are discretization-invariant because they can learn from and evaluate functions which are discretized in an arbitrary way. Since parameters are learned directly in Fourier space, resolving the functions in physical space simply amounts to projecting on the basis $e^{2\pi i \langle x, k \rangle}$ which are well-defined everywhere on $\mathbb{R}^d$. This allows zero-shot super-resolution without losing accuracy [4]. Such property provides free training acceleration and significant memory saving by training the model on the downsampled data.

Since the attention block, as shown in Fig.5.2, adopts 1×1 convolution, it retains the FNO mesh-invariant property. In this section, we benchmark the zero-shot super-resolution performance of two models. We generate data on grid size of 256×256 at the Reynolds number of 10^5 , then train both models on the downsampled data of grid size 64×64 . The trained models are evaluated at the same time step ($t=15$), on grid resolution of 64×64 , 128×128 and 256×256 , respectively.

Fig.5.17 shows the snapshots of predicted vorticity and the absolute errors at different grid resolutions. Unlike Figs.5.5 and 5.10 where the errors increase significantly with time steps and Reynolds numbers, the error rates of both models do not increase with grid size.

Fig.5.18 compares the average relative error on different resolutions: both models have con-

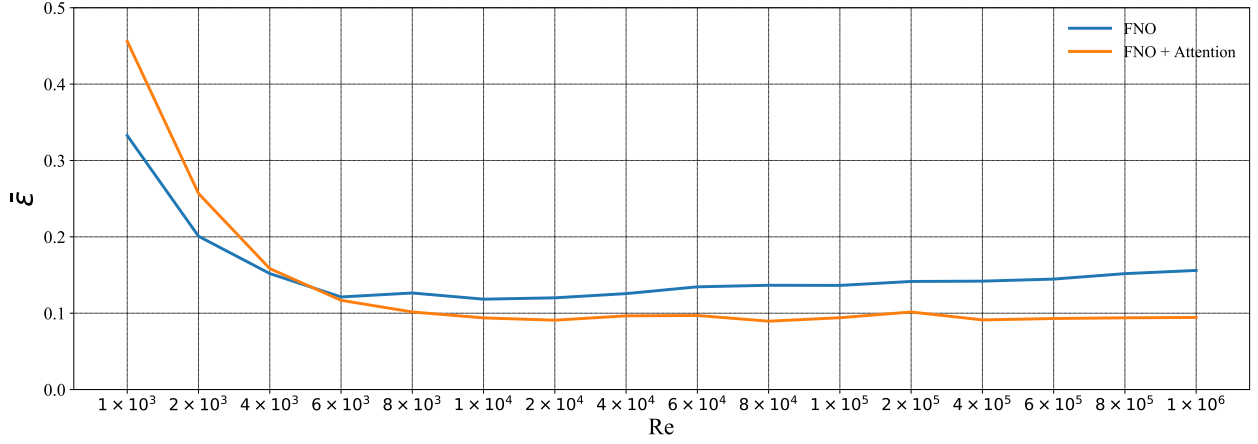


Figure 5.15 Generalization error at different Reynolds numbers (models trained on $Re = 1 \times 10^5$ dataset).

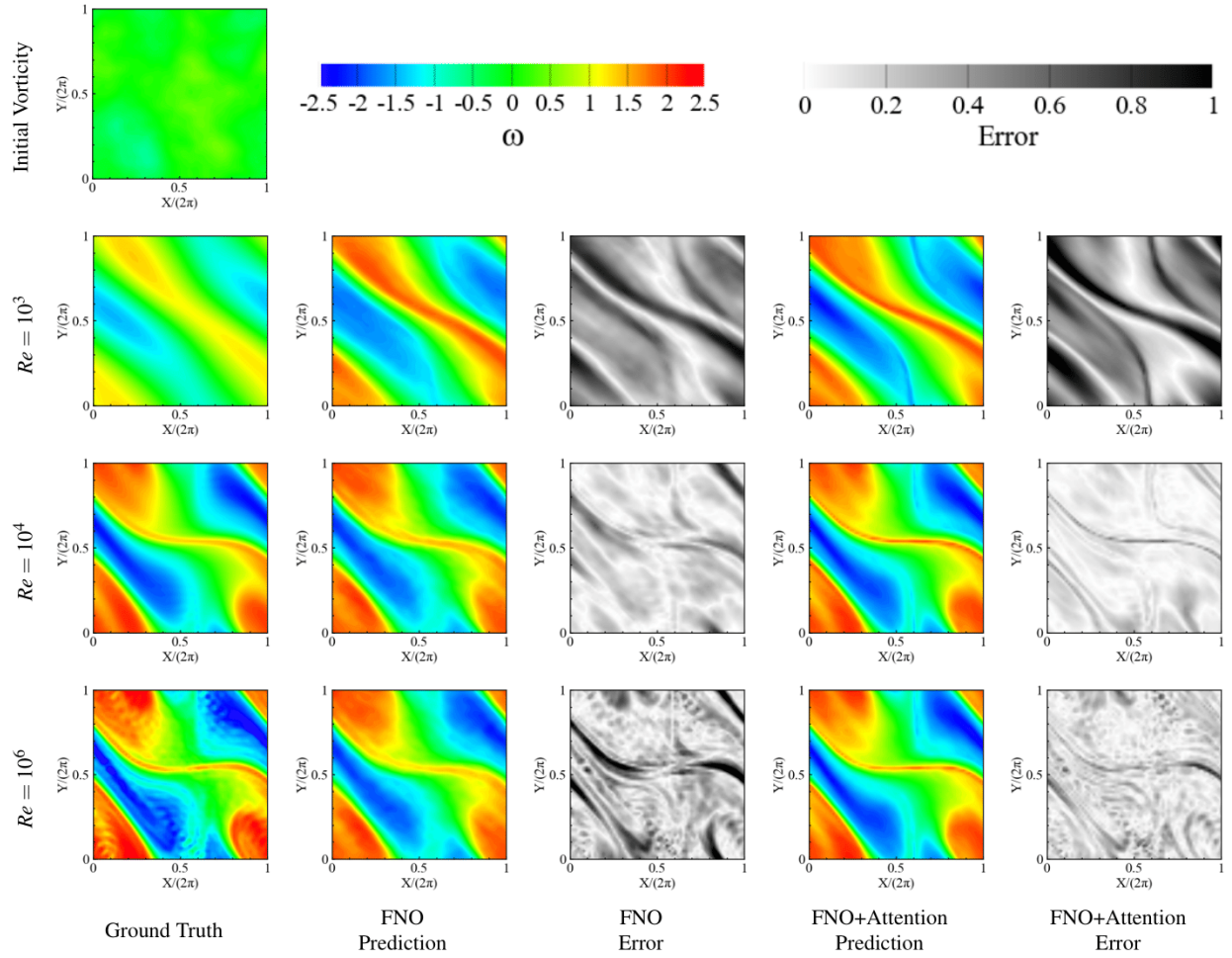


Figure 5.16 Vorticity prediction and absolute error at different Reynolds numbers. (models trained on $Re = 1 \times 10^5$ dataset).

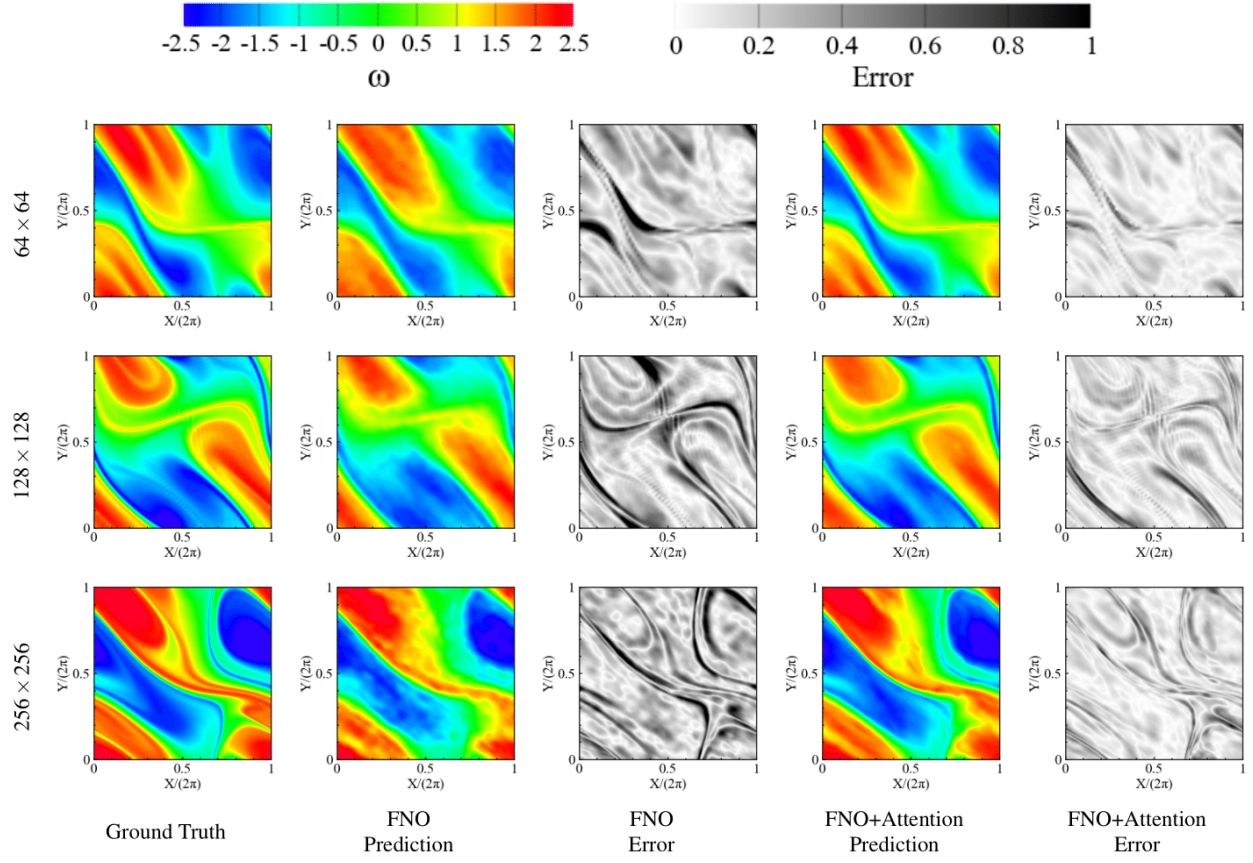


Figure 5.17 Vorticity prediction and absolute error at different resolutions.

sistent error rates among different resolutions, whereas the attention-enhanced FNO achieves over 40% error reduction at all grid resolutions.

Fig.5.19 compares the averaged vorticity spectrum of 200 test samples at different grid resolutions: the predicted vorticity spectrum of both models can agree well with the ground truth in low-wavenumber region. However, the FNO predicted spectrum deviates significantly from the ground truth at high-wavenumber region. In contrast, the FNO+Attention model can reconstruct the multi-scale flow structures accurately at all grid resolutions.

5.4.5 Performance benchmark on Taylor-Green turbulence and free shear turbulence

In this section, we benchmark the performance of different models on more challenging turbulence simulation tasks: the Taylor-Green turbulence and free shear turbulence. The Taylor-Green and free shear turbulence are governed by the same Navier-Stokes equations displayed in Eq. (5.6) without the forcing term. The Reynolds number is $Re = 10^3$ in both the Taylor-Green turbulence and free shear turbulence. The initial conditions of Taylor-Green and free shear turbulence are given by Eq.5.12 and Eq.5.13 respectively [118, 119].

$$\omega(\mathbf{x}, 0) = 2 \cos 2x_1 \cos 2x_2 + \lambda_1, \quad (5.12)$$

$$\omega(\mathbf{x}, 0) = 2\lambda_1 \cos 2x_1 + \delta\left(|x_2|, \frac{\pi}{2}\right) + \lambda_2. \quad (5.13)$$

Here, the magnitudes of perturbation λ_1 and λ_2 satisfy the Gaussian random distribution where $\lambda_1, \lambda_2 \sim \mathcal{N}(0, 10^{-3})$, and $\delta(a, b)$ is the Kronecker delta function, as described in Eq.

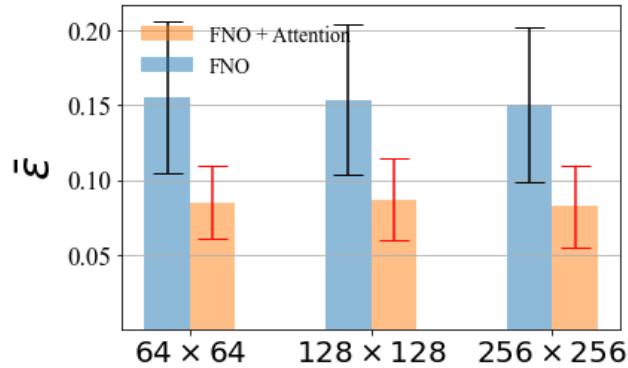


Figure 5.18 Average relative error at different resolutions.

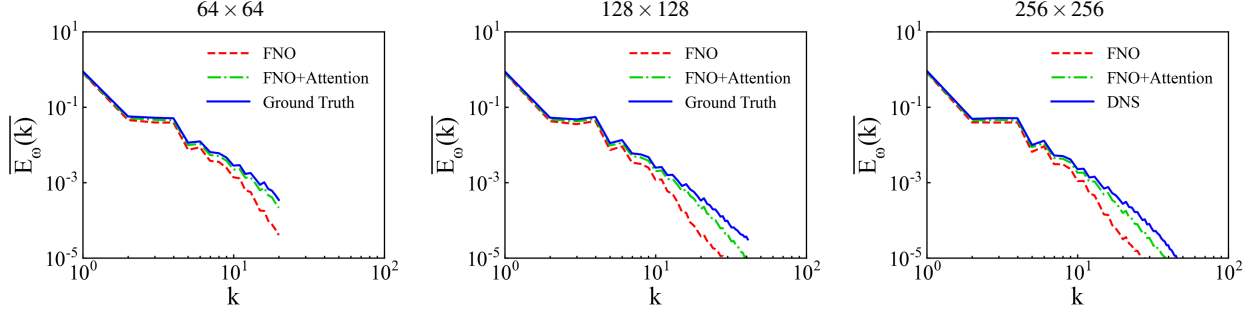


Figure 5.19 Average spectrum of vorticity at different resolutions.

5.14.

$$\delta(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases} \quad (5.14)$$

Since it takes a long time for the Taylor-Green vortex to transition from the laminar flow to turbulence, we rotate the initial Taylor-Green vortex to a certain angle ($\theta \in (-10^\circ, 10^\circ)$) to break the symmetry of the base flow [118]. We generate 1200 data samples on the grid size of 64×64 , where each sample contains 30 steps of solutions of a random initialized condition. Both models are trained on $t = 0$ to $t = 20$, and are tested on $t = 11$ to $t = 30$ to benchmark generalization performance on time dimension.

Fig. 5.20 compares the predicted vorticity and the absolute errors on a test sample of Taylor-Green turbulence. The prediction errors of both models increase with the advance of time. At $t = 11$, both models can accurately reconstruct the instantaneous spatial structures of turbulence in the beginning; at $t = 20$, the attention-enhanced FNO can still make accurate reconstructions whereas the FNO cannot; at $t = 30$ where both models are tested on unseen time steps, the errors become too large for both models to perform accurate reconstructions.

Fig.5.21 shows the ensemble-averaged vorticity spectrum $\overline{E_\omega(k)}$ of Taylor-Green flow on 200 test samples. At $t = 11$, the predicted vorticity spectrum of FNO can agree well with the ground truth at low wave numbers, but deviates from the ground truth at high wave numbers; whereas the attention-enhanced FNO can well reconstruct the vorticity spectrum at different flow scales by accurately capturing small-scale flow structures. As time advances to $t = 20$, both models deviate further away from the ground truth at high-wave number region, with attention-enhanced FNO having better agreement with the ground truth. At $t = 30$, the predicted vorticity spectrum of FNO starts to deviate from the ground truth at low-wave number region too, whereas the predicted vorticity spectrum of attention-enhanced FNO does not.

Fig.5.22 compares the predicted vorticity and the absolute errors on a test sample of the free shear turbulence. The prediction errors of both models increase with the advance of time, with the errors of attention-enhanced FNO being significantly smaller than FNO in terms of the region and magnitude. At $t = 11$ and $t = 20$, the attention-enhanced FNO captures the small-scale structures better than FNO, whereas both of them fail to reconstruct the instantaneous large-scale structures at $t = 30$.

Fig.5.23 presents the spatial-averaged relative errors of two models on predictions of the Taylor-Green turbulence and free shear turbulence. For the Taylor-Green turbulence, both models can generalize well on unseen time steps from $t = 21$ to $t = 30$; whereas for the free shear turbulence, the prediction errors increased significantly from $t = 21$ to $t = 30$, indicating that both models fail to generalize well on time dimension. In both Taylor-Green and free shear turbulence, the attention-enhanced FNO achieves 40% error reduction throughout all time steps.

5.5 Discussion

5.5.1 Why does attention work ?

One of the most common criticisms that NNs often face is the lack of interpretability, therefore they are often treated as “black-box” surrogate models. Understanding how neural network based models make inference on physical problems requires future efforts from both data scientists and specialized experts of the area. We present some thoughts and ideas from the perspective of turbulence modeling, and leave them as an open discussion.

- We have investigated the FNO feature maps $v_t(\mathbf{x})$ and the attention refined feature maps $v_t(\mathbf{x})'$. These feature maps are the output of an intermediate layer, and they contain the information of how the neural networks understand turbulence. The feature maps are stacked together and are characterized by a 3-dimensional metrics ($C \times S \times S$), where C is the channel dimension and S is the grid size. The value of the channel dimension equals to the number of convolution filters used in the Fourier layer (20 in this work). Fig.5.25 shows the unfolded feature maps of each channels. We noticed that the features learned by FNO are relatively uniformly distributed in space, whereas the attention refined features are not. Moreover, the attention refined feature maps are visually similar to the vorticity distribution, indicating that attention module can capture the turbulence nonequilibrium features better as we expected.
- Another interesting phenomenon is that the large error is closely related with the regions where the vorticity changes dramatically, as shown in Fig.5.5. We therefore investigate

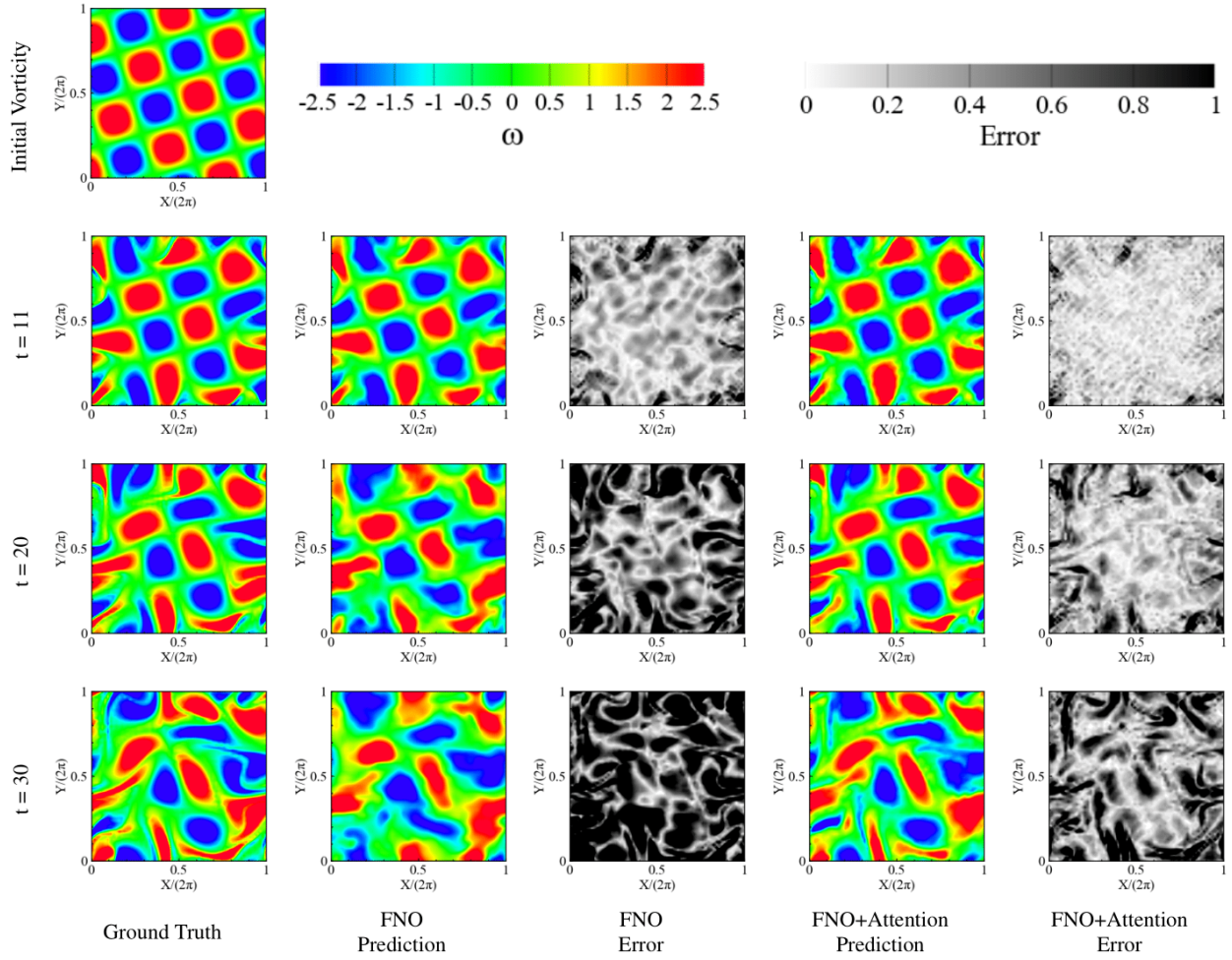


Figure 5.20 Vorticity prediction and absolute error at selected time steps (Taylor-Green turbulence).

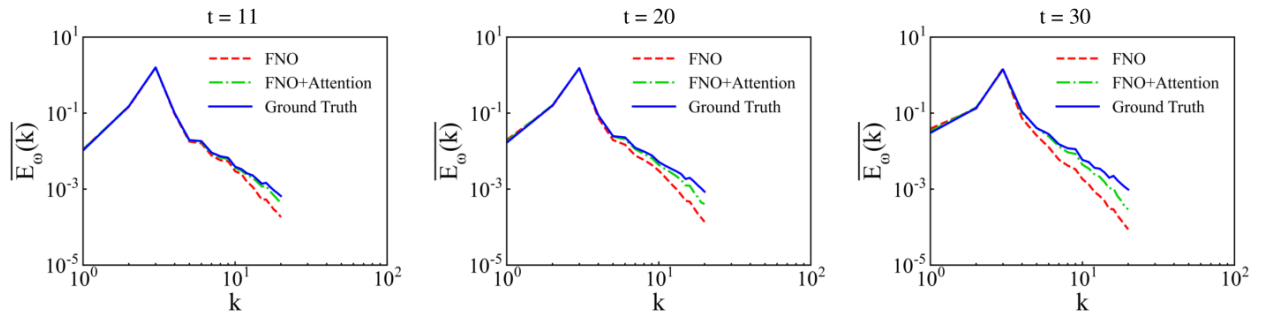


Figure 5.21 Averaged vorticity spectrum on 200 test samples (Taylor-Green turbulence).

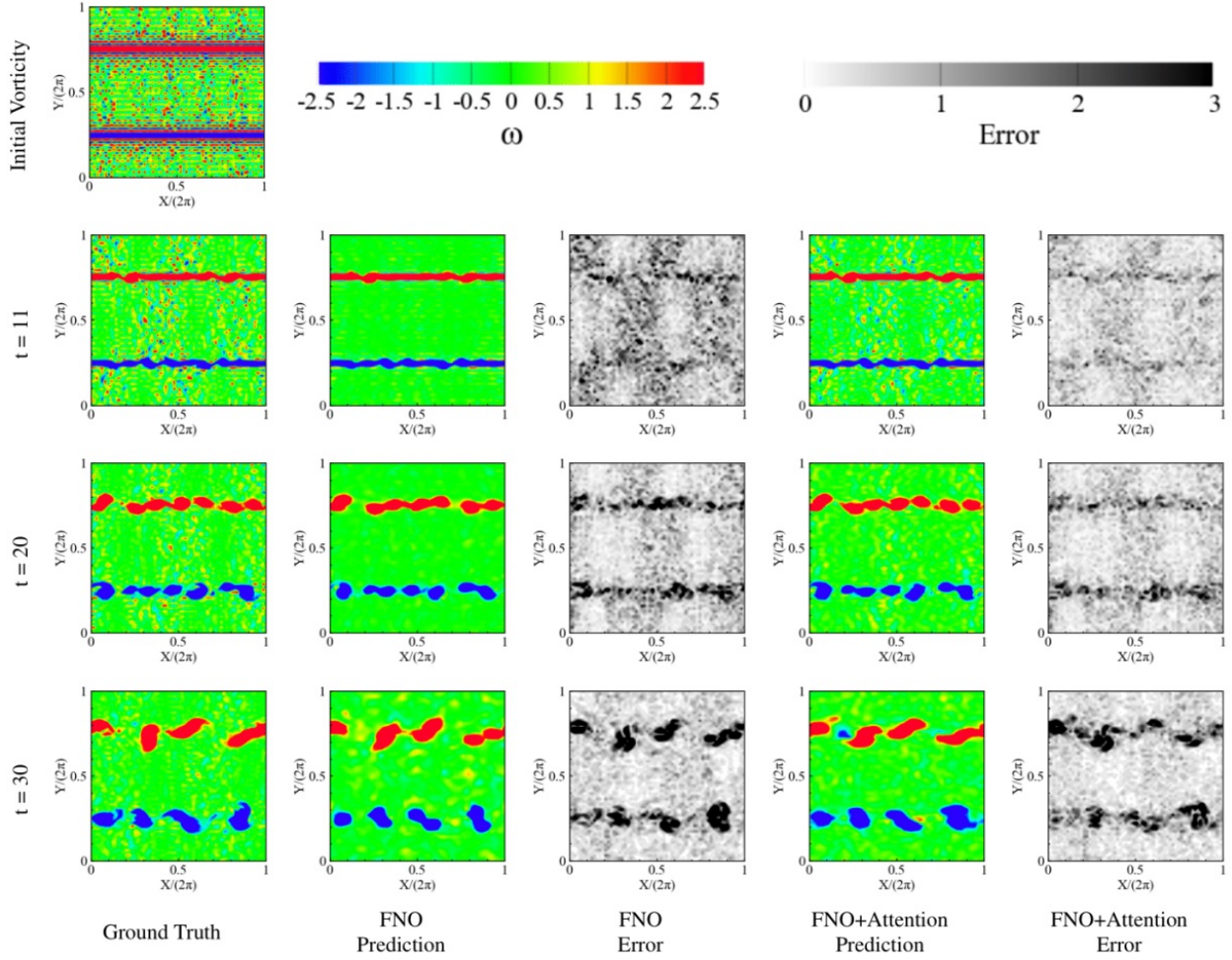


Figure 5.22 Vorticity prediction and absolute error at selected time steps (free shear turbulence).

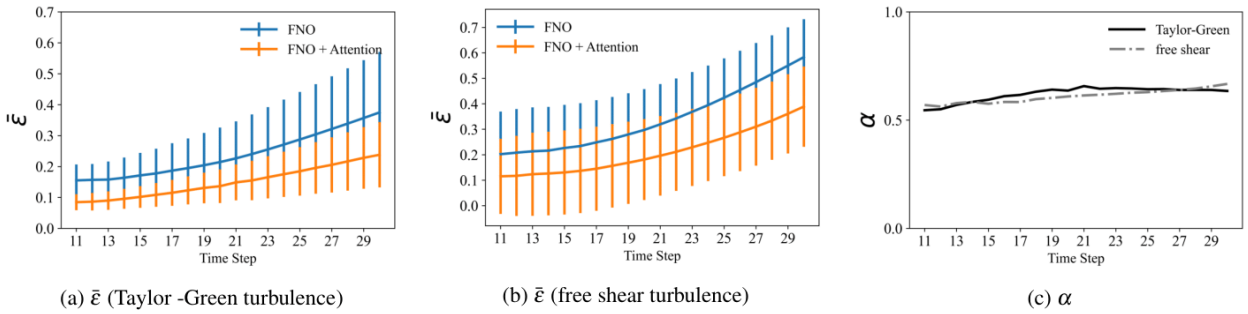


Figure 5.23 Spatial-averaged relative error comparisons on 200 test samples at consequent time steps. (a) relative error of Taylor-Green turbulence (b) relative error of free shear turbulence (c) ratios of mean relative error.

the relationship between the absolute error $|\tilde{\omega} - \omega|$ and vorticity gradient $\nabla\omega$, as shown in Fig.5.24. We noticed that the errors of both models grow with the increasing of vorticity gradient. Further improvements can be made by adding an extra vorticity gradient penalty term with the loss function during training.

5.5.2 Computational efficiency

Table 5.2 compares computational cost of 10 prediction steps on a 64×64 grid using 3 different approaches. We implement the numerical experiments on the Pytorch and MindSpore open-source deep learning frameworks. These numerical experiments are ran on a virtual machine powered by Google Colab, where the CPU type is Intel(R) Xeon(R) CPU @ 2.30GHz and GPU type is Tesla K80. Once trained, the surrogate models can be extremely efficient compared with traditional approaches. Both models provide 8000 folds speedup compared with the pseudo-spectral numerical solver. The attention-augmented neural network model achieves 40% error reduction at the same level of memory consuming and computational expense.

5.6 Conclusion

In this work, we propose an attention-enhanced neural network approach to model the nonequilibrium feature of turbulence. Numerical experiments show that: 1) the proposed model can significantly reduce the prediction error induced by temporal accumulation, and can accurately reconstruct the instantaneous spatial structures of turbulence. 2) the attention-enhanced FNO model can reduce the prediction error rate by 40% as the flow becomes more turbulent at higher Reynolds numbers. 3) the attention-enhanced model retains the mesh-invariant property of FNO: the FNO+Attention model can be trained on lower grid resolution and evaluated on higher grid resolution, without losing accuracy. 4) The attention enhanced FNO model achieves the same level of computational efficiency as compared with the original FNO model.

Table 5.2 Computational efficiency comparison of three different numerical approaches.

Method	Parameters	GPU Time (s)
Numerical Solver	N/A	502.85s
FNO	465,717	0.0579s
FNO + Attention	466,222	0.0587s

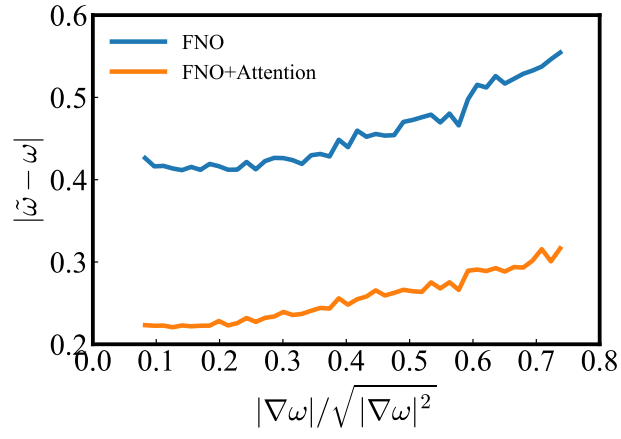


Figure 5.24 Normalized absolute error versus normalized vorticity gradient.

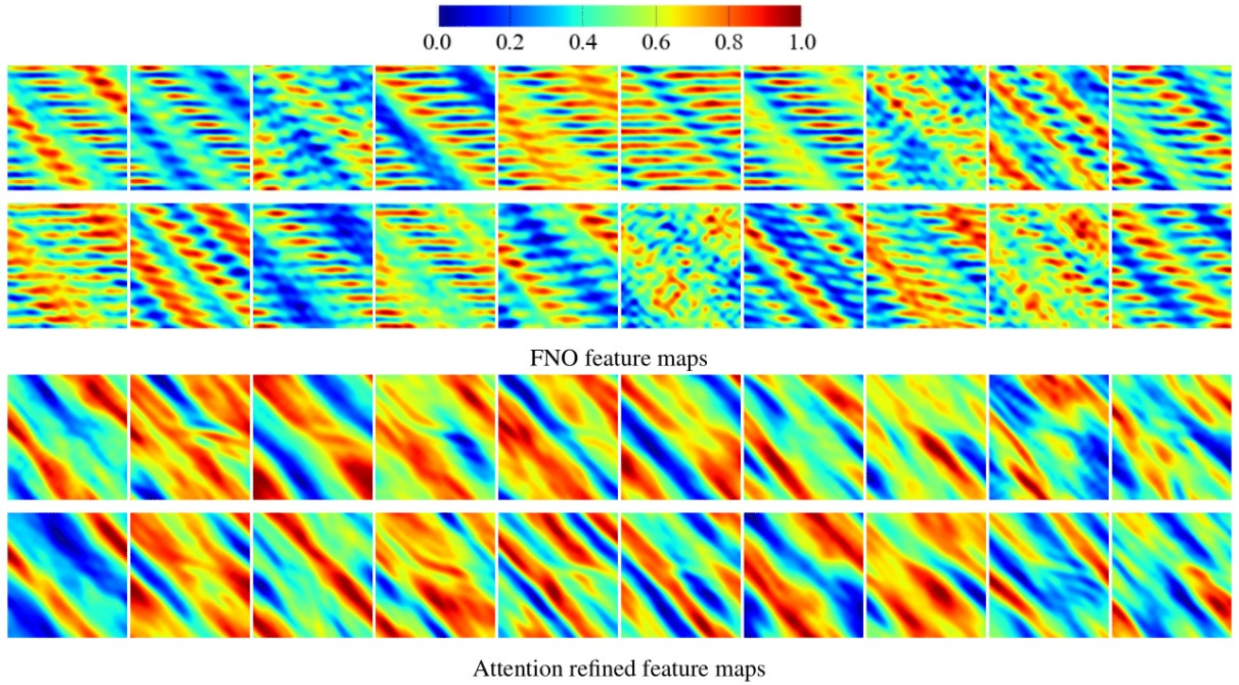


Figure 5.25 Comparison of feature maps.

CHAPTER 6 ARTICLE 3: LINEAR ATTENTION COUPLED FOURIER NEURAL OPERATOR FOR SIMULATION OF THREE-DIMENSIONAL TURBULENCE

This Chapter has been published at the Journal of Physics of Fluids on January 4th 2023, authored by Wenhui Peng, Zelong Yuan, Zhijie Li and Jianchun Wang.

6.1 Introduction

With the rising of deep learning techniques, neural networks (NNs) have been extensively explored to complement or accelerate the traditional computational fluid dynamics (CFD) modeling of turbulent flows [7, 8]. 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 [9–11].

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 [120]. 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 [90]. Some recent works aim to approximate the entire Navier-Stokes equations by deep neural networks [13–16, 121–125]. 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 [17]. Xu et al. employed the physics-informed deep learning by treating the governing equations as a parameterized constraint to reconstruct the missing flow dynamics [104]. 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 [64].

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 can not generalize well once the given equation parameters or boundary conditions change [88, 110–112]. Li et al. proposed the Fourier neural operators (FNO), which learns an entire family of partial differential equations instead of a single equation [4]. The FNO mimics the pseudo-spectral methods [105, 106]: it parameterizes the integral kernel in the Fourier space, thus directly learns the mapping from any functional parametric dependence to the solution [4]. Benefited from the expressive and efficient architecture, the FNO outperforms the previous state-of-the-art neural network models, including U-Net [75], TF-Net [64] and ResNet [117]. 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 spatio-temporal reconstruction of two-dimensional turbulent flows [19, 20, 63, 75, 76, 126–130]. These works reduce the reconstruction error mainly through adopting advanced neural network models [4, 20, 82, 131–134] or incorporating the prior physical knowledge into the model [18, 59, 64, 88, 106, 135].

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 [23, 24]. 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 [26]. Training such models can be computationally expensive because of the sheer amount of parameters involved. Further, 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 [27]. 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 [27]. The CGANs model has an acceptable accuracy in modelling the velocity features of individ-

ual snapshots of the flow, but has difficulties in modelling the probability density functions (PDFs) of the passive scalars advected with the velocity [27].

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 [77]. 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 [77], 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 utilising the convolutional operators [27]. 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 [27]. The CC-LSTM is able to predict the spatio-temporal 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 [28]. 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 [28]. 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 [29, 108, 109]. 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 [78]. 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 [78]. 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 [79]. 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 [80]. Kissas et al. coupled the attention mechanism with the neural operators towards 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 [81]. Peng et al. proposed to model the nonequilibrium feature of turbulence with the self-attention mechanism [82]. 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 [82].

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 $3D$ turbulence simulation. However, 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 [29]. 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 section 6.3.1.

Recently, Wang et al. demonstrated that the self-attention mechanism can be approximated by a low-rank matrix [30]. 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 [30]. The linear attention approximation performs on par with standard self-attention, while being much more memory and time efficient [30], 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 $3D$ turbulence simulation task.

This paper is organized as follow: section 6.2 briefly introduces the Fourier neural operator. Section 6.3 compares the standard self-attention with linear attention approximation, and introduces the detailed implementation of coupling attention with Fourier neural operator. Section 6.4 describes the details for generating $3D$ turbulence data. Section 6.5 discusses the key hyperparamters of the neural network models, including the number of input time steps and the project dimension. In section 6.6, we benchmark the prediction performance of the

original Fourier neural operator versus the linear attention coupled Fourier neural operator, via statistical and physics-based metrics. Section 6.7 compares the computational efficiency of the traditional DNS approach and the neural network models. In section 6.8 we draw conclusions.

6.2 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 [113]. The Fourier neural operators learns an approximation of $\mathcal{A} \rightarrow \mathcal{U}$ by constructing a mapping parameterized by $\theta \in \Theta$. The optimal parameters $\theta^\dagger \in \Theta$ are determined in the test-train setting by using a data-driven empirical approximation [114]. The neural operators are formulated as an iterative architecture $v_0 \mapsto v_1 \mapsto \dots \mapsto v_T$ where v_j for $j = 0, 1, \dots, T-1$ is a sequence of functions each taking values in $\mathbb{R}^{d_v}$ [17]. The FNO architecture, as shown in Fig.6.1, consists of three main steps.

Firstly, 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 $P : \mathbb{R}^{d_a} \rightarrow \mathbb{R}^{d_v}$ 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.

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. (6.1), where $\mathcal{K} : \mathcal{A} \times \Theta_{\mathcal{K}} \rightarrow \mathcal{L}(\mathcal{U}(D; \mathbb{R}^{d_v}), \mathcal{U}(D; \mathbb{R}^{d_v}))$ maps to bounded linear operators on $\mathcal{U}(D; \mathbb{R}^{d_v})$ and is parameterized by $\phi \in \Theta_{\mathcal{K}}$. Here, $W : \mathbb{R}^{d_v} \rightarrow \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is an elementally defined non-linear activation function.

$$v_{t+1}(x) = \sigma(Wv_t(x) + (\mathcal{K}(a; \phi)v_t)(x)), \quad \forall x \in D. \quad (6.1)$$

Lastly, the output $u \in \mathcal{U}$ is obtained by applying the local transformation $u(x) = Q(v_T(x))$, where $Q : \mathbb{R}^{d_v} \rightarrow \mathbb{R}^{d_u}$ 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 $f : D \rightarrow \mathbb{R}^{d_v}$ respectively. Replacing the kernel integral operator in Eq. (6.1) by a convolution operator defined in Fourier space, and applying the convolution theorem, the Fourier integral operator can be expressed by Eq. (6.2), where R_ϕ is the Fourier transform of a periodic

function $\kappa : \bar{D} \rightarrow \mathbb{R}^{d_v \times d_v}$ parameterized by $\phi \in \Theta_{\mathcal{K}}$.

$$(\mathcal{K}(\phi)v_t)(x) = \mathcal{F}^{-1}(R_\phi \cdot (\mathcal{F}v_t))(x), \quad \forall x \in D. \quad (6.2)$$

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 $k_{\max} = |Z_{k_{\max}}| = |\{k \in \mathbb{Z}^d : |k_j| \leq k_{\max,j}, \text{ for } j = 1, \dots, d\}|$. 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 $(R_\phi \cdot (\mathcal{F}v_t))_{k,l} = \sum_{j=1}^{d_v} R_{\phi k,l,j} (\mathcal{F}v_t)_{k,j}$, $k = 1, \dots, k_{\max}$, $j = 1, \dots, d_v$.

In CFD modeling, the flow is typically uniformly discretized with resolution $s_1 \times \dots \times s_d = n$, and $\mathcal{F}$ can be replaced by the fast Fourier transform (FFT). For $f \in \mathbb{R}^{n \times d_v}$, $k = (k_1, \dots, k_d) \in \mathbb{Z}_{s_1} \times \dots \times \mathbb{Z}_{s_d}$, and $x = (x_1, \dots, x_d) \in D$, the FFT $\hat{\mathcal{F}}$ and its inverse $\hat{\mathcal{F}}^{-1}$ are given by Eq. (6.3), for $l = 1, \dots, d_v$.

$$\begin{aligned} (\hat{\mathcal{F}}f)_l(k) &= \sum_{x_1=0}^{s_1-1} \dots \sum_{x_d=0}^{s_d-1} f_l(x_1, \dots, x_d) e^{-2i\pi \sum_{j=1}^d \frac{x_j k_j}{s_j}}. \\ (\hat{\mathcal{F}}^{-1}f)_l(x) &= \sum_{k_1=0}^{s_1-1} \dots \sum_{k_d=0}^{s_d-1} f_l(k_1, \dots, k_d) e^{2i\pi \sum_{j=1}^d \frac{x_j k_j}{s_j}}. \end{aligned} \quad (6.3)$$

6.3 Attention enhanced neural network

6.3.1 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 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 [29]. 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 [115]. Peng et al. proposed to couple the attention mechanism with the Fourier neural operator [82], as shown in Fig. 6.2. The architecture of the self-attention block is shown in Fig.6.3. The convolution parameters W_f, W_g, W_h learn the embedding of query, key and value, respectively, and these parameters

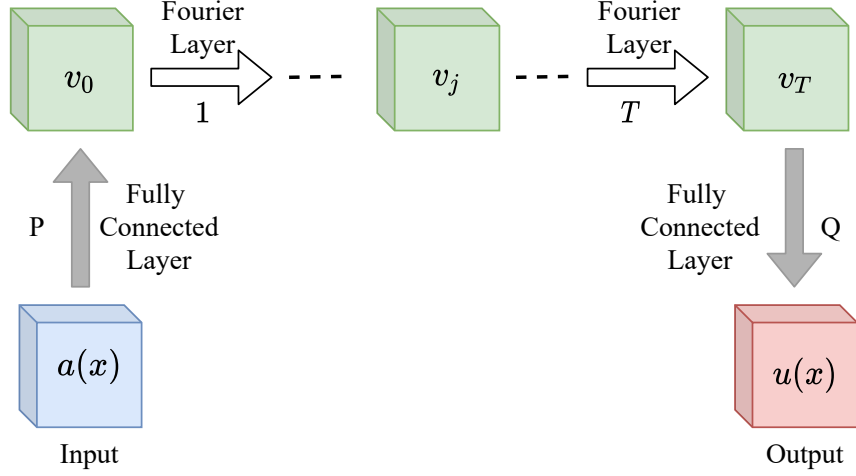


Figure 6.1 Fourier neural operator (FNO) architecture.

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. (6.4). The standard softmax function $\mathbb{R}^K \rightarrow (0, 1)^K$ is defined by the Eq. (6.5). In the 2D turbulence simulation task [82], the 2D turbulence data is generated on the grid size of 64×64 , such that $n = 64 \times 64$ and $d = 20$.

$$v_t(\mathbf{x})' = \text{Attention}(QW_f, KW_g, VW_h) = \underbrace{\text{softmax}\left[\frac{QW_f(KW_g)^T}{\sqrt{d}}\right]}_P VW_h \quad (6.4)$$

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \text{for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K \quad (6.5)$$

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 [83–85]. It is noted from Fig. 6.3 and Eq. (6.4) 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 n becomes $n = 64 \times 64 \times 64 \times 3$, 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.

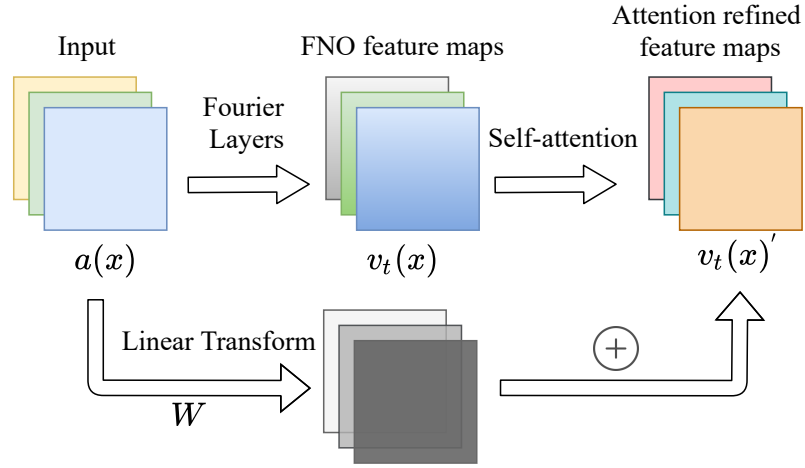


Figure 6.2 Attention enhanced Fourier neural operator.

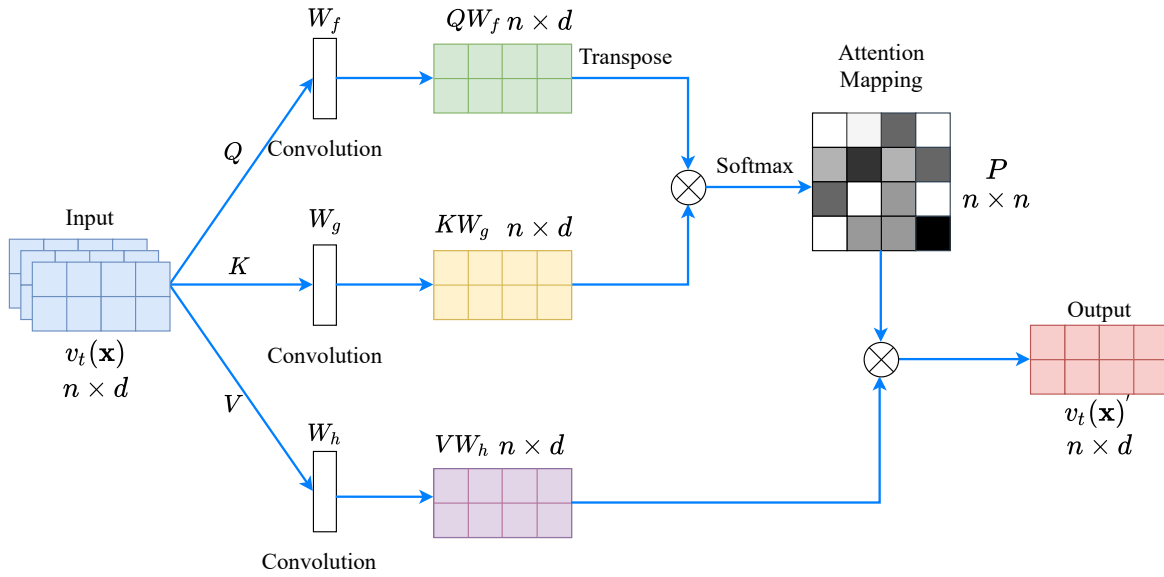


Figure 6.3 Standard self-attention module architecture.

We resolve this bottleneck with the concept of linear attention.

6.3.2 Linear attention approximation

Recently, Wang et al. showed that the attention mapping matrix P can be approximated by a low-rank matrix $\tilde{P}$ [30]: 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

$$\Pr \left(\left\| \tilde{P}w^T - Pw^T \right\| < \epsilon \left\| Pw^T \right\| \right) > 1 - o(1). \quad (6.6)$$

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.

$$P \approx P_{\text{low}} = \sum_{i=1}^k \sigma_i u_i v_i^T = \underbrace{[u_1, \dots, u_k]}_k \text{diag} \{ \sigma_1, \dots, \sigma_k \} \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} \quad (6.7)$$

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 [30]. 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 [30, 86].

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, for any row vector w of $QW_f (KW_g)^T / \sqrt{d}$, if $k = \min \{ \Theta(9d \log(d)/\epsilon^2), 5\Theta(\log(n)/\epsilon^2) \}$,

$$\Pr \left(\left\| \text{softmax}(wE^T) FVW_h - \text{softmax}(w)VW_h \right\| \leq \epsilon \left\| \text{softmax}(w) \right\| \left\| VW_h \right\| \right) > 1 - o(1) \quad (6.8)$$

The architecture of linear attention is shown in Fig. 6.4. 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 $(n \times k)$ -dimensional attention mapping matrix $\bar{P}$ using the scaled dot-product attention, as described by Eq. (6.9). 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 [30].

$$\begin{aligned}
\overline{v_t(\mathbf{x})}' &= \text{Attention}(QW_f, EKW_g, FVW_h) \\
&= \underbrace{\text{softmax}\left(\frac{QW_f(EKW_g)^T}{\sqrt{d}}\right)}_{\tilde{P}:n \times k} \cdot \underbrace{FVW_h}_{k \times d},
\end{aligned} \tag{6.9}$$

The linear self-attention module performs on par with standard self-attention module [30], 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 3D turbulence simulation task. The architecture of our proposed model, the linear attention coupled Fourier neural operator (LAFNO), is shown in Fig. 6.5. In our numerical simulation, the maximum GPU memory consumption of training LAFNO is 35.82 GB.

6.4 Dataset description

The dimensionless Navier-Stokes equations in conservative form for the 3D incompressible turbulence are given by [36]

$$\nabla \cdot \mathbf{u} = 0, \tag{6.10}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\text{Re}} \nabla^2 \mathbf{u} + \mathbf{F}, \tag{6.11}$$

where $\mathbf{u}$ is the velocity, p is the modified pressure divided by the constant density, Re is the Reynolds number, and $\mathbf{F}$ 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 [34]

$$E(k) = A_0 \left(\frac{k}{k_0}\right)^4 \exp\left[-2\left(\frac{k}{k_0}\right)^2\right], \tag{6.12}$$

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 $k_0 = 4.5786$ [34]. The vorticity $\omega = \nabla \times \mathbf{u}$ measures the local rotation of turbulent eddies, and is a Galilean-invariant variable used as the inputs and outputs of the neural networks. Fig. 6.6 shows the vorticity magnitude of ten random initial conditions, which are sampled from the training and testing datasets. Ω

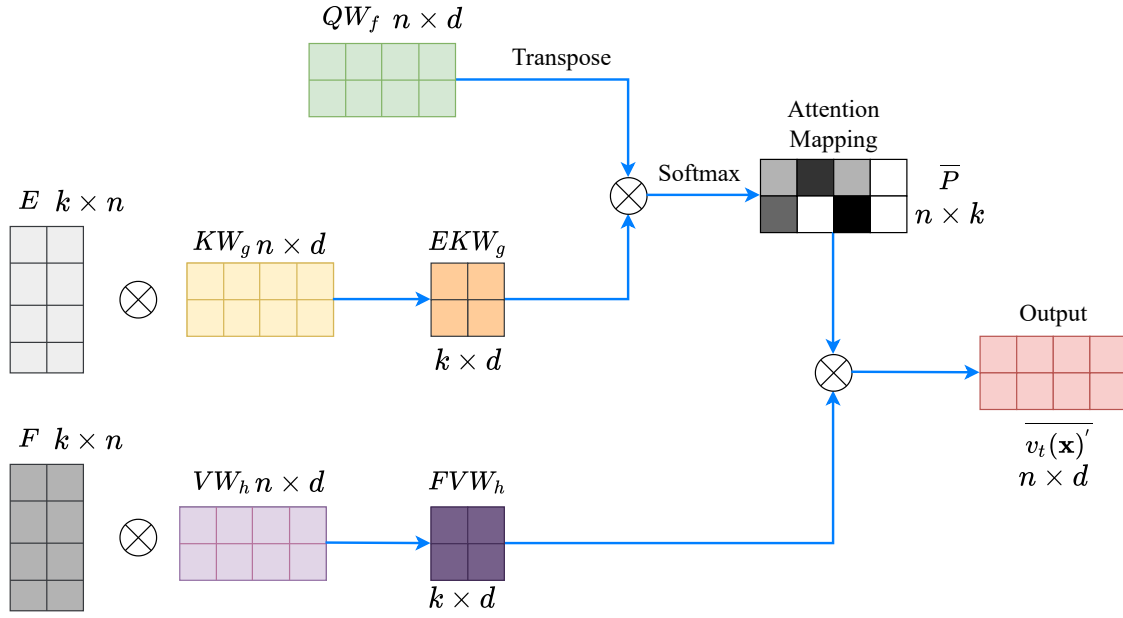


Figure 6.4 Linear self-attention module architecture.

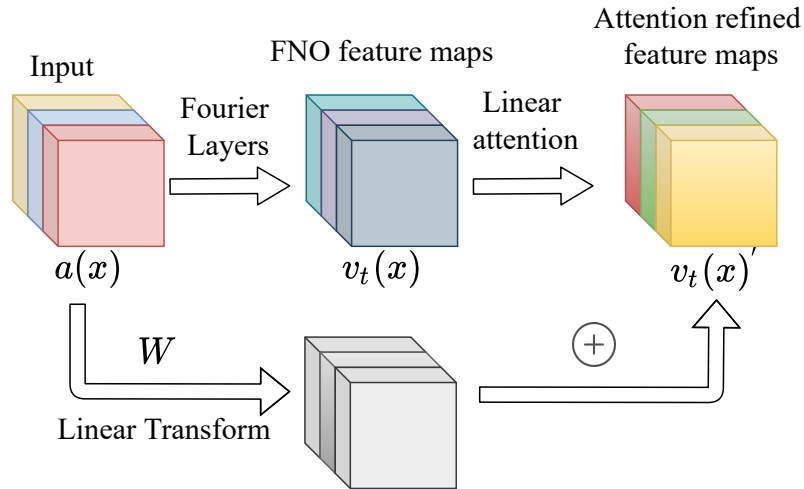


Figure 6.5 Linear attention coupled Fourier neural operator (LAFNO).

is the summation square of vorticity components, as shown in Eq.(6.13).

$$\Omega = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \quad (6.13)$$

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 [34–36]. The velocity can be expanded as the Fourier series,

$$\mathbf{u}(\mathbf{x}, t) = \sum_{\mathbf{k}} \hat{\mathbf{u}}(\mathbf{k}, t) e^{i\mathbf{k} \cdot \mathbf{x}}, \quad (6.14)$$

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 [36]

$$\mathbf{k} \cdot \hat{\mathbf{u}} = 0, \quad (6.15)$$

$$\left(\frac{d}{dt} + \frac{1}{\text{Re}} k^2 \right) \hat{\mathbf{u}}_j(\mathbf{k}, t) = -ik_l \left(\delta_{jm} - \frac{k_j k_m}{k^2} \right) \sum_{\mathbf{p}+\mathbf{q}=\mathbf{k}} \hat{\mathbf{u}}_l(\mathbf{p}, t) \hat{\mathbf{u}}_m(\mathbf{q}, t) + \hat{\mathcal{F}}_j(\mathbf{k}, t), \quad (6.16)$$

where $\mathbf{p}$ and $\mathbf{q}$ are the wavenumber vectors and k_j is the j -th component of $\mathbf{k}$. The non-local convolution sum at the right-hand side of Eq. (6.16) is introduced by the nonlinear convection term and is calculated by the pseudospectral method [36, 136]. 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 [34, 36, 136]. 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 [136]. The forced velocity $\hat{u}_j^f(\mathbf{k})$ is expressed as

$$\hat{u}_j^f(\mathbf{k}) = \alpha \hat{u}_j(\mathbf{k}), \text{ where } \alpha = \begin{cases} \sqrt{E_0(1)/E_k(1)}, & 0.5 \leq k \leq 1.5 \\ \sqrt{E_0(2)/E_k(2)}, & 1.5 \leq k \leq 2.5 \\ 1 & \text{otherwise.} \end{cases} \quad (6.17)$$

Here, $E_0(1) = 1.242477$ and $E_0(2) = 0.391356$ [136].

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 [34]. 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. (6.18), where Δt is the time step, $t_n = n\Delta t$, 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.

$$\hat{u}_j^{n+1} = \hat{u}_j^n + \Delta t \left[\frac{3}{2} \hat{R}_j(\hat{\mathbf{u}}^n, t_n) - \frac{1}{2} \hat{R}_j(\hat{\mathbf{u}}^{n-1}, t_{n-1}) \right]. \quad (6.18)$$

6.5 Hyperparameters of neural networks

6.5.1 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, C is the number of vorticity components. 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 reference [4]. It contains four Fourier layers, the number of truncated Fourier modes is 20, and the number of convolution channels is 36 [4].

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 [4, 82]. However, the input number of ten time steps is too expensive for 3D turbulence simulations, since 3D data is 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 error ϵ of next step, as shown in table 6.1. The relative error ϵ of the model is defined by Eq. (6.19), where $\tilde{\omega}$ is the predicted vorticity vector and ω is the ground truth.

$$\epsilon = \frac{\|\tilde{\omega} - \omega\|_2}{\|\omega\|_2}, \text{ where } \|\mathbf{a}\|_2 = 1/n \sqrt{\sum_{k=1}^n |\mathbf{a}_k|^2}. \quad (6.19)$$

We notice from table 6.1 that the error decreases significantly with the number of input steps

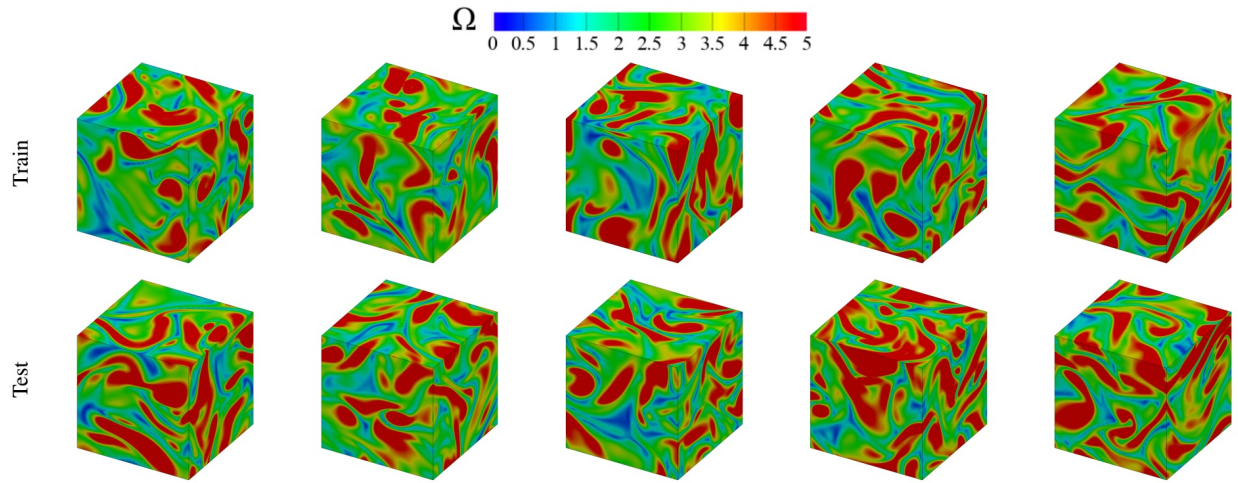


Figure 6.6 Ten random sampled initial fields of vorticity magnitude from the training set and testing set.

Table 6.1 Prediction 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

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.

6.5.2 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 can not be approximated with sufficient accuracy, which further leads to the increase of prediction error. In this work, the 3D turbulence data is 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 6.2 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 $k \geq d$.

6.6 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

Table 6.2 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

vorticity at previous 4 time steps solutions as input, and gives the vorticity at the next time step as output. During training, the vorticity of first 4 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. We use 2400 samples for training and 600 samples for testing. The learning curve (testing error) is shown in Fig.6.7. 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 [34]. 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 $\Delta T/\tau = 1$.

Fig. 6.8 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.

Fig. 6.9 shows a 2D vorticity slice in the middle of Z axis from the same test sample in Fig. 6.8. 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.

Fig.6.10 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 1% 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 54% and 32% respectively. It is also noticed that the LAFNO achieves 40% error reduction compared with FNO, throughout all time steps.

Fig.6.11 compares the ensemble-averaged velocity spectrum $\overline{E(k)}$ using 100 test samples. The velocity field is calculated from vorticity field by solving the Poisson equation $\nabla^2 \mathbf{u} =$

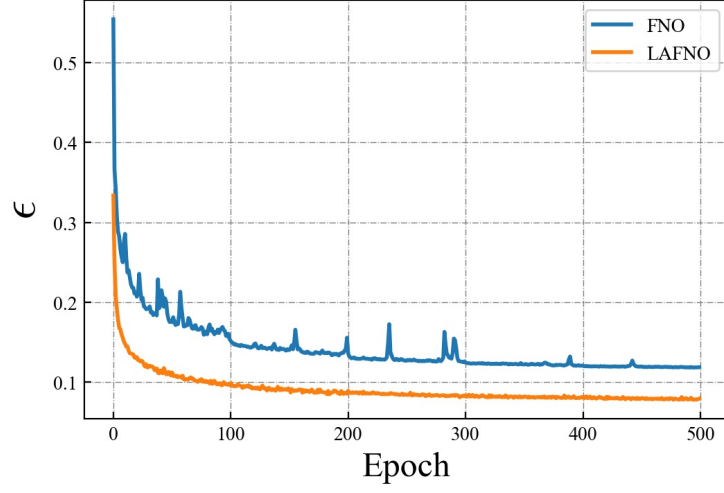


Figure 6.7 Learning curve (testing error) of FNO and LAFNO

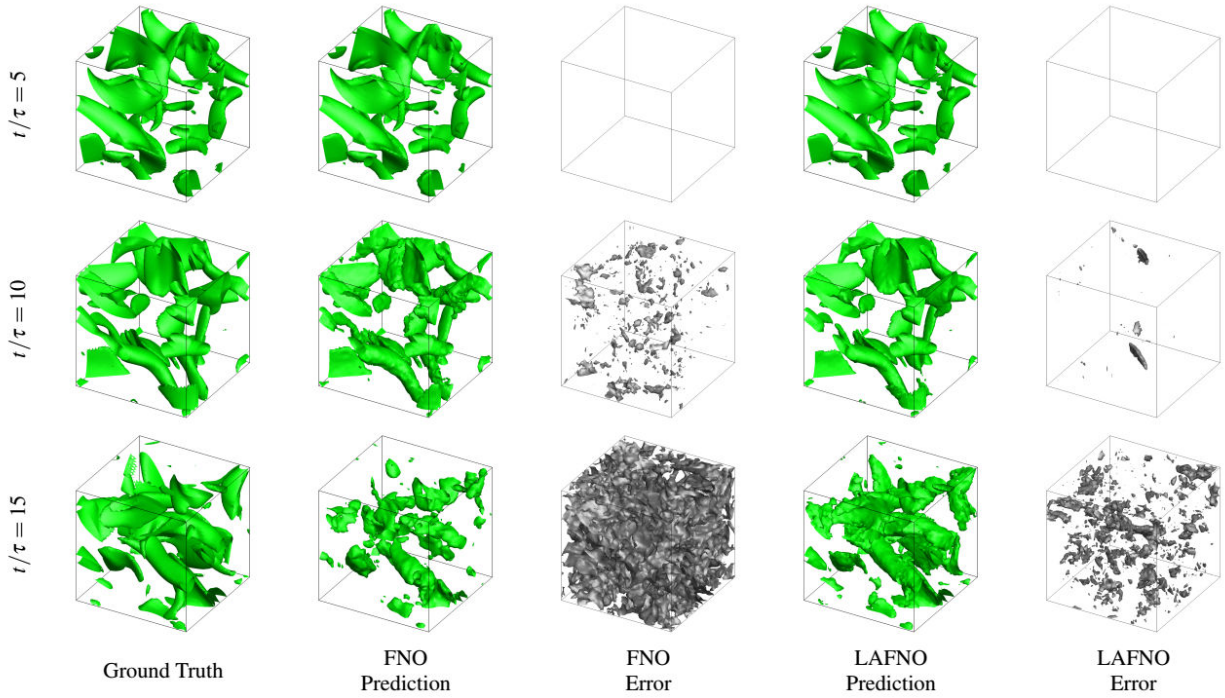


Figure 6.8 Isosurfaces of the normalized vorticity magnitude $\Omega/\Omega^{rms} = 1.5$ and relative error $\epsilon = 0.5$ at selected time steps.

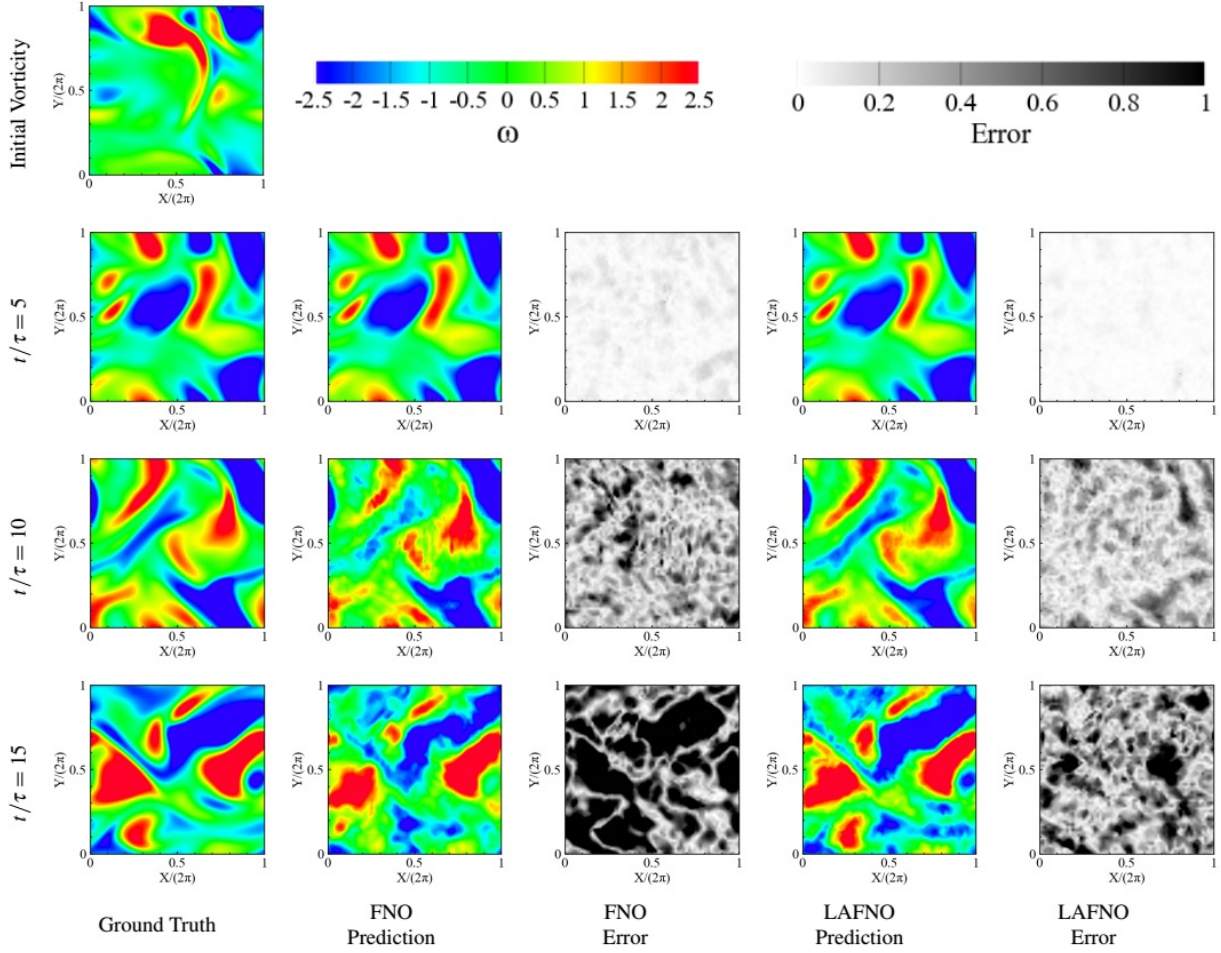
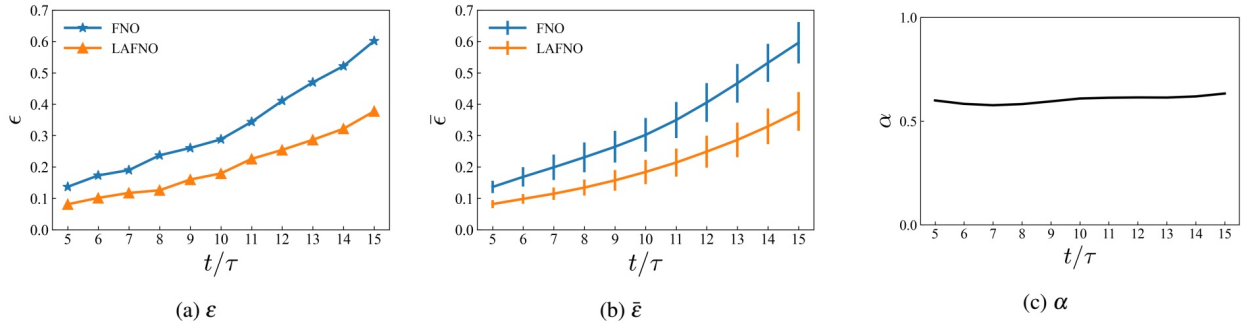


Figure 6.9 Vorticity (2D slice) prediction and absolute error at selected time steps.



(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 $\alpha = \bar{\epsilon}_{LAFNO} / \bar{\epsilon}_{FNO}$.

Figure 6.10 Relative error comparison at consequent time steps

$-\nabla \times \omega$. 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 can not.

Fig.6.12 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.

6.7 Computational efficiency

Table 6.3 compares computational cost of 10 prediction steps on a $64 \times 64 \times 64$ grid using 3 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 8350C @ 2.60GHz. 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. Both neural network models provides 20 folds speedup compared with the DNS approach with the traditional numerical solver. Moreover, the LAFNO

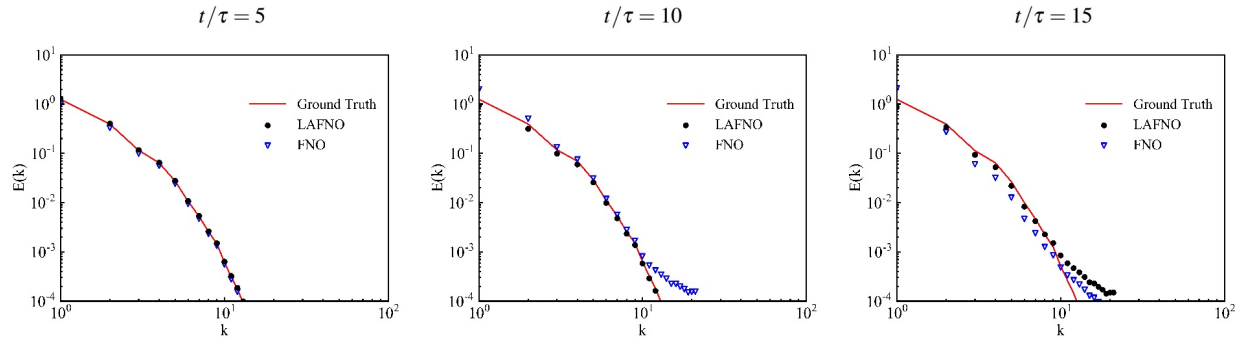


Figure 6.11 Averaged velocity spectrum on 100 test samples.

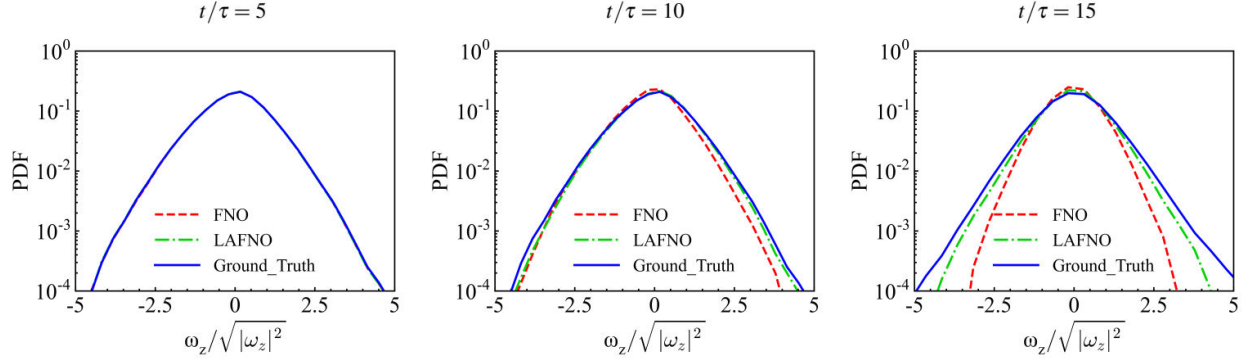


Figure 6.12 PDFs of the normalized vorticity component ω_z at different time steps.

achieves 40% error reduction at the same level of memory consuming and computational expense as compared to FNO.

6.8 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.

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 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

Table 6.3 Computational efficiency comparison of three different numerical approaches.

Method	Parameters	Time (s)
Numerical Solver	N/A	12.53s
LAFNO	70,570,874	0.576s
FNO	69,605,997	0.538s

problems with high-dimensional data.

CHAPTER 7 GENERAL DISCUSSION

The synthetic data used in Chapter 4 is generated with the Xfoil solver [33]. Since the flow is in the compressible regime, and the angle of attack is within the linear range (-2° to 10°), the Xfoil is accurate [137]. The synthetic data used in Chapter 5 and 6 is generated by solving the Navier-Stokes equations 5.6. Since the synthesized data is the solution of PDEs, there is no noise involved.

Both the FNO model and FNO+Attention model are not sensitive to initial conditions, as reported in section 5.4, they can achieve the same level of prediction accuracy when tested for different initial conditions than the training set. However, both models are sensitive to the change of Reynolds number, as reported in section 5.4. Fig.5.15 shows the generalization errors of FNO and FNO+Attention models at different Reynolds numbers: both models are trained on $Re = 10^5$ dataset, and tested on unseen Reynolds numbers ranging from $Re = 10^3$ to $Re = 10^6$. It is noted that the FNO generalizes better at low Reynolds numbers (below $Re = 5 \times 10^3$), whereas the FNO+Attention generalizes better at high Reynolds numbers.

The main purpose of this thesis is to improve the prediction accuracy over existing state of the art neural network models for incompressible fluid simulations. However, the acceptable error rate for practical applications varies depending on the specific task. General applications such as weather forecast and city wind simulation usually have higher acceptable error rate than medical applications such as blood fluid simulations.

It is worth noting that the third contribution of this thesis is limited the idea of FNO+Attention. Since 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. The third contribution resolve this bottleneck by coupling the FNO with linear attention. 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, thus allowing the attention mechanism to be coupled with neural networks and trained on GPUs.

The FNO model is discretization-invariant because the parameters are learned directly in Fourier space, resolving the functions in physical space simply amounts to projecting on the basis $e^{2\pi i \langle x, k \rangle}$ which are well-defined everywhere on $\mathbb{R}^d$. Such property allows FNO to be trained on lower resolution and tested on higher resolution without having to retrain the model. Since the attention mechanism adopts 1×1 convolution, as shown in Fig.5.2, the FNO+Attention model retains the mesh-invariant property. However, change of geometry

(e.g. changing the geometry from cubic box to sphere) will lead to change of input. In such cases, both the FNO and FNO+Attention model have to be retrained.

CHAPTER 8 CONCLUSION

8.1 Summary of works

In this thesis, we explored the applications of deep learning techniques on the simulation of fluid dynamic systems.

In the first part, we introduced a novel neural network architecture, the Element Spatial Convolutional Neural Network (ESCNN), to predict the aerodynamic coefficients of airfoils. The ESCNN outperforms existing state-of-the-art NNs in terms of prediction accuracy, with two orders of less parameters. Instead of processing the airfoils as images, the ESCNN adopts a physics-informed structural input. We provided insight interpretation from the physical perspective as to why ESCNN achieves better performance than state-of-the-art AeroCNN. We investigated a layer of the ESCNN model during the training process, and discovered that the NN learns patterns that emerge from aerodynamics, and such patterns are clearly reflected within one of the layer of the NN. We show that the ESCNN learns the physical laws and equations of aerodynamics, and make inference based on the learned physical knowledge.

In the second part, we proposed to couple the self-attention layer with the Fourier neural operator to accelerate the simulation of high-Reynolds-number turbulent flow. We compare the model performance against the state-of-the-art neural network model as the baseline, on two dimensional turbulence prediction task. Numerical experiments show that the attention-enhanced neural network model outperforms existing state-of-the-art baselines, and can accurately reconstruct a variety of statistics and instantaneous spatial structures of turbulence at high Reynolds numbers. Our research in this part has its practical usage: our proposed attention-enhanced model provides 40% error reduction with 1% increase in parameters, at the same level of computational cost, in the high-Reynolds-number turbulent flow prediction task.

In the third part, we extend the self-attention mechanism to three-dimensional turbulence simulation task. 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. We resolve this issue with the concept of 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,

thus allowing the attention mechanism to be coupled with neural networks and trained on GPUs. Numerical simulations show that the linear attention mechanism provides 40% error reduction at the same level of computational cost.

8.2 Objective review

The main purpose of this thesis is to improve the prediction accuracy over existing state of the art neural network models for incompressible fluid simulations. We cover three typical kinds of fluid simulations problems with increasing level of complexity: (1) Two-dimensional inviscid flow, where no viscosity is involved. (2) Two-dimensional viscous turbulent flow. (3) Three-dimensional viscous turbulent flow.

In chapter 4 we propose the ESCNN over the baseline AeroCNN, towards the airfoil lift prediction in two-dimensional inviscid flow. The ESCNN achieved 70% error reduction compared with the baseline AeroCNN. In chapter 5 we propose attention-enhanced FNO over the baseline FNO, towards the flow field prediction task in two-dimensional turbulent flow. The attention-enhanced FNO achieved 40% error reduction compared with the baseline FNO. In chapter 6 we propose linear attention coupled FNO over the baseline FNO, towards the flow field prediction task in three-dimensional turbulent flow. The linear attention coupled FNO achieved 40% error reduction compared with the baseline FNO.

8.3 Limits and future research

The first limitation is that the ESCNN model for airfoil lift coefficient prediction task is trained and validated on the laminar flow. However, calculating the airfoil lift coefficient in the turbulent range is more often required in engineering applications. It is necessary to explore the feasibility and performance of ESCNN in the turbulent range in future work.

The second limitation is that our interpretation towards ESCNN is based on the existing laminar flow theory. Interpreting how the neural networks make inference in turbulent flow is much more challenging since existing understanding towards turbulence theory can not provide sufficient understanding as the laminar flow theory.

The third limitation is that despite that the attention mechanism has been shown to be effective in reducing the prediction error of FNO, the problem of error accumulation over time is improved but not fundamentally resolved. To solve this problem in future work, physical constraints can be incorporated to ensure that the predictions are subject to the governing equations and conservation laws [18, 59, 88, 106, 135, 138].

The fourth limitation is that the attention mechanism is coupled with FNO only in real space in this work. It is worth investigating the effect of incorporating the attention block inside the Fourier layer in complex space, and effect of combination attention in both the real space and the complex space.

The fourth 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) [139], physics-informed neural operator (PINO) [18], adaptive fourier neural operators (AFNO) [130] and U-shaped neural operators (UNO) [140] have been proposed to simulate complex 2D flows. Li et al. proposed the Geo-FNO [133], 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. These FNO variants, including the Geo-FNO, could be extended and coupled with the attention mechanism to effectively simulate the 3D engineering complex flows in future work.

REFERENCES

- [1] Y. Zhang, W. J. Sung, and D. N. Mavris, “Application of convolutional neural network to predict airfoil lift coefficient,” in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018, p. 1903.
- [2] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.
- [3] J. D. Anderson Jr, *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 2010.
- [4] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, “Fourier neural operator for parametric partial differential equations,” *arXiv preprint arXiv:2010.08895*, 2020.
- [5] J. N. Kutz, “Deep learning in fluid dynamics,” *Journal of Fluid Mechanics*, vol. 814, pp. 1–4, 2017.
- [6] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [7] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, “Machine learning for fluid mechanics,” *Annual Review of Fluid Mechanics*, vol. 52, pp. 477–508, 2020.
- [8] K. Duraisamy, G. Iaccarino, and H. Xiao, “Turbulence modeling in the age of data,” *Annual Review of Fluid Mechanics*, vol. 51, pp. 357–377, 2019.
- [9] J. Ling, A. Kurzawski, and J. Templeton, “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance,” *Journal of Fluid Mechanics*, vol. 807, pp. 155–166, 2016.
- [10] R. Maulik, O. San, A. Rasheed, and P. Vedula, “Subgrid modelling for two-dimensional turbulence using neural networks,” *Journal of Fluid Mechanics*, vol. 858, pp. 122–144, 2019.
- [11] A. Beck, D. Flad, and C.-D. Munz, “Deep neural networks for data-driven les closure models,” *Journal of Computational Physics*, vol. 398, p. 108910, 2019.

- [12] G. D. Portwood, B. T. Nadiga, J. A. Saenz, and D. Livescu, “Interpreting neural network models of residual scalar flux,” *Journal of Fluid Mechanics*, vol. 907, 2021.
- [13] B. Lusch, J. N. Kutz, and S. L. Brunton, “Deep learning for universal linear embeddings of nonlinear dynamics,” *Nature communications*, vol. 9, no. 1, pp. 1–10, 2018.
- [14] J. Sirignano and K. Spiliopoulos, “Dgm: A deep learning algorithm for solving partial differential equations,” *Journal of computational physics*, vol. 375, pp. 1339–1364, 2018.
- [15] H. Tang, L. Li, M. Grossberg, Y. Liu, Y. Jia, S. Li, and W. Dong, “An exploratory study on machine learning to couple numerical solutions of partial differential equations,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 97, p. 105729, 2021.
- [16] 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, 2020, pp. 352–372.
- [17] 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 preprint arXiv:2003.03485*, 2020.
- [18] 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 preprint arXiv:2111.03794*, 2021.
- [19] V. Sekar, Q. Jiang, C. Shu, and B. C. Khoo, “Fast flow field prediction over airfoils using deep learning approach,” *Physics of Fluids*, vol. 31, no. 5, p. 057103, 2019.
- [20] J. Chen, E. Hachem, and J. Viquerat, “Graph neural networks for laminar flow prediction around random two-dimensional shapes,” *Physics of Fluids*, vol. 33, no. 12, p. 123607, 2021.
- [21] R. Han, Y. Wang, Y. Zhang, and G. Chen, “A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network,” *Physics of Fluids*, vol. 31, no. 12, p. 127101, 2019.
- [22] H. Wang, Z. Yang, B. Li, and S. Wang, “Predicting the near-wall velocity of wall turbulence using a neural network for particle image velocimetry,” *Physics of Fluids*, vol. 32, no. 11, p. 115105, 2020.

- [23] M. Momenifar, E. Diao, V. Tarokh, and A. D. Bragg, “Dimension reduced turbulent flow data from deep vector quantisers,” *Journal of Turbulence*, pp. 1–33, 2022.
- [24] A. Glaws, R. King, and M. Sprague, “Deep learning for in situ data compression of large turbulent flow simulations,” *Physical Review Fluids*, vol. 5, no. 11, p. 114602, 2020.
- [25] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian, “Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast,” *arXiv preprint arXiv:2211.02556*, 2022.
- [26] F. Juefei-Xu, V. Naresh Boddeti, and M. Savvides, “Local binary convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 19–28.
- [27] A. T. Mohan, D. Tretiak, M. Chertkov, and D. Livescu, “Spatio-temporal deep learning models of 3d turbulence with physics informed diagnostics,” *Journal of Turbulence*, vol. 21, no. 9-10, pp. 484–524, 2020.
- [28] 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,” *Physics of Fluids*, vol. 33, no. 2, p. 025116, 2021.
- [29] 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*, 2017, pp. 5998–6008.
- [30] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, “Linformer: Self-attention with linear complexity,” *arXiv preprint arXiv:2006.04768*, 2020.
- [31] M. Selig, *UIUC airfoil data site*. Department of Aeronautical and Astronautical Engineering University of Illinois at Urbana-Champaign, 1996.
- [32] W. Song and A. Keane, “A study of shape parameterisation methods for airfoil optimisation,” in *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2004, p. 4482.
- [33] M. Drela, “Xfoil: An analysis and design system for low reynolds number airfoils,” in *Low Reynolds number aerodynamics*. Springer, 1989, pp. 1–12.

- [34] Z. Yuan, Y. Wang, C. Xie, and J. Wang, “Dynamic iterative approximate deconvolution models for large-eddy simulation of turbulence,” *Physics of Fluids*, vol. 33, no. 8, p. 085125, 2021.
- [35] 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,” *Journal of Fluid Mechanics*, vol. 883, 2020.
- [36] S. B. Pope and S. B. Pope, *Turbulent flows*. Cambridge university press, 2000.
- [37] J. Hermann, Z. Schätzle, and F. Noé, “Deep-neural-network solution of the electronic schrödinger equation,” *Nature Chemistry*, vol. 12, no. 10, pp. 891–897, 2020.
- [38] A. Davies, P. Veličković, L. Buesing, S. Blackwell, D. Zheng, N. Tomašev, R. Tanburn, P. Battaglia, C. Blundell, A. Juhász *et al.*, “Advancing mathematics by guiding human intuition with ai,” *Nature*, vol. 600, no. 7887, pp. 70–74, 2021.
- [39] T. McGrath, A. Kapishnikov, N. Tomašev, A. Pearce, D. Hassabis, B. Kim, U. Paquet, and V. Kramnik, “Acquisition of chess knowledge in alphazero,” *arXiv preprint arXiv:2111.09259*, 2021.
- [40] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland *et al.*, “Improved protein structure prediction using potentials from deep learning,” *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.
- [41] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, “Machine learning and the physical sciences,” *Reviews of Modern Physics*, vol. 91, no. 4, p. 045002, 2019.
- [42] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” *arXiv preprint arXiv:1802.03426*, 2018.
- [43] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [44] M. Brenner, J. Eldredge, and J. Freund, “Perspective on machine learning for advancing fluid mechanics,” *Physical Review Fluids*, vol. 4, no. 10, p. 100501, 2019.
- [45] A. P. Singh, S. Medida, and K. Duraisamy, “Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils,” *AIAA Journal*, pp. 2215–2227, 2017.

- [46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [47] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: analysis, applications, and prospects,” *IEEE transactions on neural networks and learning systems*, 2021.
- [48] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, “Recent advances in convolutional neural networks,” *Pattern recognition*, vol. 77, pp. 354–377, 2018.
- [49] A. Ajit, K. Acharya, and A. Samanta, “A review of convolutional neural networks,” in *2020 international conference on emerging trends in information technology and engineering (ic-ETITE)*. IEEE, 2020, pp. 1–5.
- [50] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A convolutional neural network for modelling sentences,” *arXiv preprint arXiv:1404.2188*, 2014.
- [51] M. Lin, Q. Chen, and S. Yan, “Network in network,” *arXiv preprint arXiv:1312.4400*, 2013.
- [52] R. Thirumalainambi and J. Bardina, “Training data requirement for a neural network to predict aerodynamic coefficients,” in *Independent Component Analyses, Wavelets, and Neural Networks*, vol. 5102. SPIE, 2003, pp. 92–103.
- [53] G. Sun and S. Wang, “A review of the artificial neural network surrogate modeling in aerodynamic design,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 16, pp. 5863–5872, 2019.
- [54] M. M. Rai and N. K. Madavan, “Application of artificial neural networks to the design of turbomachinery airfoils,” *Journal of Propulsion and Power*, vol. 17, no. 1, pp. 176–183, 2001.
- [55] Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, and Y. Chen, “Convolutional recurrent neural networks: Learning spatial dependencies for image representation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 18–26.
- [56] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, “Convolutional learning of spatio-temporal features,” in *European conference on computer vision*. Springer, 2010, pp. 140–153.

- [57] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [58] W. Peng, Y. Zhang, and M. Desmarais, “Spatial convolution neural network for efficient prediction of aerodynamic coefficients,” in *AIAA Scitech 2021 Forum*, 2021, p. 0277.
- [59] 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,” *Journal of Computational Physics*, vol. 426, p. 109951, 2021.
- [60] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, “Machine learning–accelerated computational fluid dynamics,” *Proceedings of the National Academy of Sciences*, vol. 118, no. 21, 2021.
- [61] J. Ren, H. Wang, G. Chen, K. Luo, and J. Fan, “Predictive models for flame evolution using machine learning: A priori assessment in turbulent flames without and with mean shear,” *Physics of Fluids*, vol. 33, no. 5, p. 055113, 2021.
- [62] X. Jin, P. Cheng, W.-L. Chen, and H. Li, “Prediction model of velocity field around circular cylinder over various reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder,” *Physics of Fluids*, vol. 30, no. 4, p. 047105, 2018.
- [63] C. Cheng and G.-T. Zhang, “Deep learning method based on physics informed neural network with resnet block for solving fluid flow problems,” *Water*, vol. 13, no. 4, p. 423, 2021.
- [64] 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*, 2020, pp. 1457–1466.
- [65] Y. Li, J. Chang, Z. Wang, and C. Kong, “An efficient deep learning framework to reconstruct the flow field sequences of the supersonic cascade channel,” *Physics of Fluids*, vol. 33, no. 5, p. 056106, 2021.
- [66] A. Galassi, M. Lippi, and P. Torroni, “Attention in natural language processing,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4291–4308, 2020.
- [67] K. Cho, A. Courville, and Y. Bengio, “Describing multimedia content using attention-based encoder-decoder networks,” *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1875–1886, 2015.

- [68] F. Wang and D. M. Tax, “Survey on the attention based rnn model and its applications in computer vision,” *arXiv preprint arXiv:1601.06823*, 2016.
- [69] V. Mnih, N. Heess, A. Graves *et al.*, “Recurrent models of visual attention,” *Advances in neural information processing systems*, vol. 27, 2014.
- [70] J. Ba, V. Mnih, and K. Kavukcuoglu, “Multiple object recognition with visual attention,” *arXiv preprint arXiv:1412.7755*, 2014.
- [71] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, “Draw: A recurrent neural network for image generation,” in *International Conference on Machine Learning*. PMLR, 2015, pp. 1462–1471.
- [72] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [73] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” *Advances in neural information processing systems*, vol. 29, 2016.
- [74] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *arXiv preprint arXiv:1710.10196*, 2017.
- [75] J. Chen, J. Viquerat, and E. Hachem, “U-net architectures for fast prediction of incompressible laminar flows,” *arXiv preprint arXiv:1910.13532*, 2019.
- [76] 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,” *Physics of Fluids*, vol. 33, no. 12, p. 125119, 2021.
- [77] 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,” *Advances in neural information processing systems*, vol. 28, 2015.
- [78] P. Wu, S. Gong, K. Pan, F. Qiu, W. Feng, and C. Pain, “Reduced order model using convolutional auto-encoder with self-attention,” *Physics of Fluids*, vol. 33, no. 7, p. 077107, 2021.
- [79] I. K. Deo and R. Jaiman, “Learning wave propagation with attention-based convolutional recurrent autoencoder net,” *arXiv preprint arXiv:2201.06628*, 2022.

- [80] Q. Liu, W. Zhu, F. Ma, X. Jia, Y. Gao, and J. Wen, “Graph attention network-based fluid simulation model,” *AIP Advances*, vol. 12, no. 9, p. 095114, 2022.
- [81] G. Kissas, J. H. Seidman, L. F. Guilhoto, V. M. Preciado, G. J. Pappas, and P. Perdikaris, “Learning operators with coupled attention,” *Journal of Machine Learning Research*, vol. 23, no. 215, pp. 1–63, 2022.
- [82] W. Peng, Z. Yuan, and J. Wang, “Attention-enhanced neural network models for turbulence simulation,” *Physics of Fluids*, vol. 34, no. 2, p. 025111, 2022.
- [83] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, “Training with quantization noise for extreme model compression,” *arXiv preprint arXiv:2004.07320*, 2020.
- [84] R. Child, S. Gray, A. Radford, and I. Sutskever, “Generating long sequences with sparse transformers,” *arXiv preprint arXiv:1904.10509*, 2019.
- [85] N. Kitaev, Ł. Kaiser, and A. Levskaya, “Reformer: The efficient transformer,” *arXiv preprint arXiv:2001.04451*, 2020.
- [86] W. B. Johnson, “Extensions of Lipschitz mappings into a Hilbert space,” *Contemp. Math.*, vol. 26, pp. 189–206, 1984.
- [87] N. B. Erichson, M. Muehlebach, and M. W. Mahoney, “Physics-informed autoencoders for lyapunov-stable fluid flow prediction,” *arXiv preprint arXiv:1905.10866*, 2019.
- [88] 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,” *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.
- [89] G. Pang, L. Lu, and G. E. Karniadakis, “fpinns: Fractional physics-informed neural networks,” *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2603–A2626, 2019.
- [90] X. Yang, S. Zafar, J.-X. Wang, and H. Xiao, “Predictive large-eddy-simulation wall modeling via physics-informed neural networks,” *Physical Review Fluids*, vol. 4, no. 3, p. 034602, 2019.
- [91] S. Wang, Y. Teng, and P. Perdikaris, “Understanding and mitigating gradient flow pathologies in physics-informed neural networks,” *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

- [92] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, “Physics-informed neural networks for high-speed flows,” *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020.
- [93] M. Ding, Z. Chen, T. Du, P. Luo, J. Tenenbaum, and C. Gan, “Dynamic visual reasoning by learning differentiable physics models from video and language,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [94] Y. Zhang, P. Tiño, A. Leonardis, and K. Tang, “A survey on neural network interpretability,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.
- [95] Q. Zhang and S.-C. Zhu, “Visual interpretability for deep learning: a survey,” *arXiv preprint arXiv:1802.00614*, 2018.
- [96] Q. Zhang, Y. N. Wu, and S.-C. Zhu, “Interpretable convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8827–8836.
- [97] S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao *et al.*, “Interpretability of deep learning models: A survey of results,” in *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smart-world/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)*. IEEE, 2017, pp. 1–6.
- [98] J. Franklin, “The elements of statistical learning: data mining, inference and prediction,” *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.
- [99] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” *arXiv preprint arXiv:1505.00853*, 2015.
- [100] Y. Wang, Z. Yuan, C. Xie, and J. Wang, “Artificial neural network-based spatial gradient models for large-eddy simulation of turbulence,” *AIP Advances*, vol. 11, no. 5, p. 055216, 2021.
- [101] J. Park and H. Choi, “Toward neural-network-based large eddy simulation: application to turbulent channel flow,” *Journal of Fluid Mechanics*, vol. 914, 2021.
- [102] D. Schmidt, R. Maulik, and K. Lyras, “Machine learning accelerated turbulence modeling of transient flashing jets,” *Physics of Fluids*, vol. 33, no. 12, p. 127104, 2021.

- [103] S. Pawar, O. San, A. Nair, A. Rasheed, and T. Kvamsdal, “Model fusion with physics-guided machine learning: Projection-based reduced-order modeling,” *Physics of Fluids*, vol. 33, no. 6, p. 067123, 2021.
- [104] H. Xu, W. Zhang, and Y. Wang, “Explore missing flow dynamics by physics-informed deep learning: The parameterized governing systems,” *Physics of Fluids*, vol. 33, no. 9, p. 095116, 2021.
- [105] Y. Fan, C. O. Bohorquez, and L. Ying, “Bcr-net: A neural network based on the nonstandard wavelet form,” *Journal of Computational Physics*, vol. 384, pp. 1–15, 2019.
- [106] 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.
- [107] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [108] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, “Image transformer,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 4055–4064.
- [109] X. Liu and M. Milanova, “Visual attention in deep learning: a review,” *Int Rob Auto J*, vol. 4, no. 3, pp. 154–155, 2018.
- [110] S. Pan and K. Duraisamy, “Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability,” *SIAM Journal on Applied Dynamical Systems*, vol. 19, no. 1, pp. 480–509, 2020.
- [111] K. Wu and D. Xiu, “Data-driven deep learning of partial differential equations in modal space,” *Journal of Computational Physics*, vol. 408, p. 109307, 2020.
- [112] H. Xu, D. Zhang, and J. Zeng, “Deep-learning of parametric partial differential equations from sparse and noisy data,” *Physics of Fluids*, vol. 33, no. 3, p. 037132, 2021.
- [113] B. Beauzamy, *Introduction to Banach spaces and their geometry*. Elsevier, 2011.
- [114] V. N. Vapnik, “An overview of statistical learning theory,” *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

- [115] G. J. Kowalski, *Information retrieval systems: theory and implementation*. Springer, 2007, vol. 1.
- [116] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [117] 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*, 2016, pp. 770–778.
- [118] O. San and R. Maulik, “Stratified Kelvin–Helmholtz turbulence of compressible shear flows,” *Nonlinear Processes in Geophysics*, vol. 25, no. 2, pp. 457–476, 2018.
- [119] T. K. Sengupta, N. Sharma, and A. Sengupta, “Non-linear instability analysis of the two-dimensional Navier-Stokes equation: The Taylor-Green vortex problem,” *Physics of Fluids*, vol. 30, no. 5, p. 054105, 2018.
- [120] 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,” *Journal of Computational Physics*, vol. 458, p. 111090, 2022.
- [121] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, “Neural operator: Learning maps between function spaces,” *arXiv preprint arXiv:2108.08481*, 2021.
- [122] X. Meng, L. Yang, Z. Mao, J. del Águila Ferrandis, and G. E. Karniadakis, “Learning functional priors and posteriors from data and physics,” *Journal of Computational Physics*, vol. 457, p. 111073, 2022.
- [123] 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 preprint arXiv:2205.08304*, 2022.
- [124] S. Goswami, K. Kontolati, M. D. Shields, and G. E. Karniadakis, “Deep transfer learning for partial differential equations under conditional shift with deepnet,” *arXiv preprint arXiv:2204.09810*, 2022.
- [125] A. A. Howard, M. Perego, G. E. Karniadakis, and P. Stinis, “Multifidelity deep operator networks,” *arXiv preprint arXiv:2204.09157*, 2022.

- [126] 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 Dynamics Research*, vol. 52, no. 6, p. 065501, 2020.
- [127] ———, “Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes,” *Theoretical and Computational Fluid Dynamics*, vol. 34, no. 4, pp. 367–383, 2020.
- [128] A. Patil, J. Viquerat, and E. Hachem, “Autoregressive transformers for data-driven spatio-temporal learning of turbulent flows,” *arXiv preprint arXiv:2209.08052*, 2022.
- [129] 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,” *Advances in Water Resources*, vol. 163, p. 104180, 2022.
- [130] J. Guibas, M. Mardani, Z. Li, A. Tao, A. Anandkumar, and B. Catanzaro, “Adaptive Fourier neural operators: Efficient token mixers for transformers,” *arXiv preprint arXiv:2111.13587*, 2021.
- [131] Z. Li, K. Meidani, and A. B. Farimani, “Transformer for partial differential equations’ operator learning,” *arXiv preprint arXiv:2205.13671*, 2022.
- [132] X. Ye, H. Li, P. Jiang, T. Wang, and G. Qin, “Learning transient partial differential equations with local neural operators,” *arXiv preprint arXiv:2203.08145*, 2022.
- [133] Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar, “Fourier neural operator with learned deformations for PDEs on general geometries,” *arXiv preprint arXiv:2207.05209*, 2022.
- [134] A. F. Psaros, K. Kawaguchi, and G. E. Karniadakis, “Meta-learning PINN loss functions,” *Journal of Computational Physics*, vol. 458, p. 111121, 2022.
- [135] S. Goswami, A. Bora, Y. Yu, and G. E. Karniadakis, “Physics-informed deep neural operators networks,” *arXiv preprint arXiv:2207.05748*, 2022.
- [136] Z. Yuan, C. Xie, and J. Wang, “Deconvolutional artificial neural network models for large eddy simulation of turbulence,” *Physics of Fluids*, vol. 32, no. 11, p. 115106, 2020,11,1.
- [137] K. Kallstrom, “Exploring airfoil table generation using xfoil and overflow,” *NASA Technical Report*, 2022.

- [138] 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 preprint arXiv:2202.05476*, 2022.
- [139] A. Tran, A. Mathews, L. Xie, and C. S. Ong, “Factorized fourier neural operators,” *arXiv preprint arXiv:2111.13802*, 2021.
- [140] M. A. Rahman, Z. E. Ross, and K. Azizzadenesheli, “U-no: U-shaped neural operators,” *arXiv preprint arXiv:2204.11127*, 2022.