공학박사 학위논문

Application of Machine Learning Technique to Generate Flow Properties in a Round Turbulent Jet

원형 난류 제트 내 흐름 특성 생성을 위한 기계학습기법의 적용

2023 년 8 월

서울대학교 대학원

건설환경공학부

최성은

Application of Machine Learning Technique to Generate Flow Properties in a Round Turbulent Jet

원형 난류 제트 내 흐름 특성 생성을 위한
기계학습기법의 적용

지도 교수 황 진 환
이 논문을 공학박사 학위논문으로 제출함

2023 년 3 월

서울대학교 대학원
건설환경공학부
최 성 은

최성은의 박사학위논문을 인준함
2023 년 6 월

위 원 장 박 용 성 (인)

부위원장 황 진 환 (인)

위 원 손 상 영 (인)

위 원 김 열 우 (인)

위 원 박 인 환 (인)

Abstract

Application of Machine Learning Technique to Generate Flow Properties in a Round Turbulent Jet

by

Seongeun Choi

Doctor of Philosophy in Civil and Environmental Engineering

Seoul National University

Professor Jin Hwan Hwang, Advisor

This study utilized Convolutional Neural Network (CNN) and Generative Adversarial Network (GAN) algorithms to generate flow velocity and turbulent properties in jet flows and evaluate their performance. Three-dimensional Large Eddy Simulation (LES) data sets

of a round turbulent jet flow, obtained through the Computational Fluid Dynamics (CFD) software, OpenFOAM, were used as ground-truth to train the Machine Learning (ML) algorithms. The good agreement between the simulation results and observation data from laboratory experiments confirmed the reliability of the LES data set. For this study, the resolved data, excluding the subgrid-scale (SGS) part of the LES data, served as the input data. Subsequently, a ML model was employed to generate the velocity field, incorporating the SGS component. The primary focus was on predicting the SGS information using the input data and producing a complete velocity field with accurate fluid flow characteristics. Various grid sizes and Reynolds numbers were used to assess the performance of the CNN and GAN algorithms.

The CNN algorithm was tested with two architectures. The first architecture consists of layers connected in one path, while the second architecture has layers connected in three paths. The second architecture outperformed the first architecture in terms of generating flow velocity and turbulent kinetic energy. It demonstrated exceptional extrapolation capabilities, accurately reproducing flow properties regardless of the grid size and Reynolds number of the input data considered in this study. Similarly, the GAN algorithm was tested with two architectures. The first architecture had fully connected layers, while the second architecture had convolutional layers. The second architecture outperformed the first architecture in reproducing the velocity field. However, both GAN architectures faced challenges in reproducing turbulent attributes and spatial correlations, especially being influenced by the grid size and Reynolds number.

Furthermore, this study investigates the accuracy of CNN techniques in simulating turbulence characteristics using time-averaged velocity data. The input data was time-averaged velocity, and the output data predicted turbulent kinetic energy (TKE), time-averaged production, and dissipation. The CNN model was trained and tested using different Reynolds numbers for TKE, time-averaged production, and dissipation. The results showed that the CNN model reasonably predicted the trend of the ground truth. Overall, the CNN algorithms were found to have the potential to accurately predict turbulence characteristics using time-averaged velocity data. The findings in this study can provide valuable insights for selecting an appropriate ML algorithm to reproduce complex turbulent flow phenomena.

Keyword: Turbulent Jet; Convolutional Neural Network; Generative Adversarial Network; Large Eddy Simulation; OpenFOAM.

Student number: 2017-24760

CONTENTS

List of Figures

List of Tables

CHAPTER 1. INTRODUCTION

1.1 Background and Necessity

In hydraulic and aerodynamic engineering, the turbulent jet has been observed in numerous environmental problems. From a hydraulic perspective, these environmental problems can be related to the discharge of water waste in domestic, agricultural, and industrial areas, including waste disposal and treatment facilities. They can also encompass chemical release accidents caused by marine traffic, offshore exploration, and the construction of marine structures. Additionally, the plunging jet aerator serves as a representative example of using multiple jets to add and mix oxygen into liquids by incorporating air bubbles in hydraulic engineering (Kumar et al., 2022). The use of multiple jets helps in increasing the mixing efficiency of chemical components and reducing the concentration of contaminants. However, reproducing such multiple jets or analyzing their efficiency through numerical simulations can be time-consuming, and the accuracy of the simulated results may not be guaranteed, depending on the grid resolution. In this context, it is necessary to propose appropriate machine learning (ML) models to address the aforementioned issues. The first step in this process should be to develop ML algorithms that can accurately predict the flow of a single jet before tackling multiple jet flows.

To study complex jet flows, the various methods have been utilized, including theoretical analysis (Fischer et al., 1979; Lee et al., 2003), laboratory experiments (Xu & Antonia, 2002; Matsuda & Sakakibara, 2005), and numerical investigations (Bogey et al., 2005; Kim et al., 2009; Huang et al., 2020; Xiao et al., 2021), to investigate the characteristics of jet flow. Fischer et al. (1979) have pointed out that turbulent jet velocity profiles are influenced by factors such as the surrounding fluid conditions, the shape of the jet inlet, and the presence of buoyancy force. They have provided equations for time-averaged profiles based on these conditions. In particular, Lee et al. (2003) have focused on distinguishing between jet and plume and have mathematically modeled the velocity profiles and momentum flux of planar and spherical jets. While time-averaged profiles were relatively

straightforward to formulate theoretically, accurately describing instantaneous turbulent flows considering various flow and geometric parameters posed a challenge. Several laboratory experiments were conducted to describe turbulent jet velocity profiles by considering multiple parameters simultaneously. Xu and Antonia (2002) conducted experiment to investigate the impact of jet inlet shape. They compared fully developed turbulent profiles of a smooth contraction nozzle and a pipe nozzle jet flow. Matsuda and Sakakibara (2005) performed experiments at various Reynolds numbers to study three-dimensional vortex structures in jet flows. However, the experimental data collected in these studies were limited in terms of the covered area and duration of measurements. Furthermore, altering the jet's shape, inclination, or ambient flow conditions proved to be difficult in these experiments. Due to the limitations associated with laboratory experiments, numerical simulations have been widely utilized as an alternative approach. Numerical simulations provided greater flexibility in modifying flow and geometric parameters, as well as in obtaining data for analysis.

In the early studies of simulating jet flows, the Reynolds-Averaged Navier-Stokes (RANS) turbulence model was commonly employed (Georgiadis et al., 2006; Faghani et al., 2011; Wang et al., 2014). Georgiadis et al. (2006) examined various RANS models and compared the numerical results with laboratory experiments. They found that the k-epsilon model demonstrated the best agreement with the experimental data compared to other models. Subsequently, Faghani et al. (2011) conducted numerical simulations using the k-epsilon turbulence model for different nozzle shapes. They analyzed parameters such as mean velocity, turbulence intensity, and turbulence kinetic energy of the jet flow. However, RANS turbulence models only provided ensemble-averaged values and cannot capture the variation in flow velocity over time. To address this limitation, the Large Eddy Simulation (LES) turbulence model was introduced. LES can provide instantaneous flow velocity and pressure over time, as well as replicate the anisotropy of turbulence (Dejoan & Leschziner, 2005; Gohil et al., 2014). Dejoan and Leschziner (2005) performed LES simulations that were in good agreement with experimental measurements. They obtained turbulence energy budgets and Reynolds stresses based on the validated

model performance. Gohil et al. (2014) implemented higher-order spatial and temporal discretization schemes, enabling the capture of more detailed turbulent flow fields compared to Dejoan and Leschziner (2005). However, it was important to note that in order to obtain high-quality numerical results, the numerical grid size should be sufficiently small to properly model the shear layer and near-wall turbulence eddies. Otherwise, the reliability of the numerical results can be compromised (Weaver & Mišković, 2021). LES simulations can outperform RANS simulations only when an adequate mesh resolution is employed and appropriate wall-damping methods are selected.

Wang et al. (2010) and Taub et al. (2013) employed Direct Numerical Simulation (DNS) as an optimal model to address the challenges discussed earlier. DNS is a computational method that directly calculates the flow equations without any turbulence model assumptions. It offers excellent numerical accuracy and provides detailed information about turbulent statistics. In their study, Wang et al. (2010) conducted DNS of a turbulent jet flow at a Reynolds number of 4,700. They verified that various turbulent statistics, including turbulence intensity and Reynolds stress, matched well with experimental values. Taub et al. (2013) focused on investigating mean velocity and turbulence properties, including vortical behaviors near the jet inlet, using DNS. They directly calculated turbulence statistics without relying on assumptions made in previous studies. This approach provided more accurate results. However, it is important to note that employing DNS in most cases, especially at high Reynolds numbers, can be computationally intensive and may not be feasible due to the requirement for extensive computational resources. In the study by Wang et al. (2010), over 20 million grids were utilized for the Reynolds number of 4,700, and Taub et al. (2013) required over 34 million grids for the Reynolds number of 2,000. While DNS offers excellent accuracy, its application is typically limited to lower Reynolds numbers or cases where computational resources are available to handle significant computational demands.

ML has been employed to improve the accuracy and efficiency of numerical simulations in turbulence modeling. ML is a popular technique for developing turbulence models (Ling et al., 2016;

Wu et al., 2017; Gamahara & Hattori, 2017; Park & Choi, 2021). Ling et al. (2016) used the ML model to develop a nonlinear eddy viscosity model for the RANS model. The proposed model demonstrated sufficient accuracy in predicting Reynolds stresses, but it struggled with mean velocity predictions. To enhance the model's performance, Wu et al. (2017) modified the turbulence model by separating the Reynolds stress into linear and nonlinear components, resulting in accurate mean velocity field results. In the LES model, Gamahara and Hattori (2017) aimed to improve the existing subgrid-scale (SGS) eddy viscosity models using an artificial neural network (ANN). They developed a function that related the resolved scales to the SGS stress tensor without assuming isotropic turbulence. The stress tensor's individual components were trained separately, avoiding any isotropic turbulence assumptions. While this model generally provided a good fit, it had the limitation of being applicable only within a certain grid size range. Park and Choi (2021) proposed another SGS eddy viscosity model using various neural networks for a turbulent channel flow. They predicted and compared the SGS stresses using different prediction models. Their results indicated that neural network models can also be applied to SGS. However, these models were restricted by specific geometries and a certain range of Reynolds numbers.

This study proposes a new approach that use ML models based only on the resolved flow properties quantified by an LES model. The SGS eddy behaviors are substituted with those calculated by a pre-trained ML model for various Reynolds numbers. This approach allows for the use of lower mesh resolutions compared to DNS, resulting in reduced computational costs and improved accuracy in SGS modeling. Since the existing SGS turbulence models are not utilized in this study, the ML models are used as the alternative methods. More specifically, among ML models, convolutional neural network (CNN) and generative adversarial network (GAN) are used for generating the SGS turbulence. CNN is a neural network model primarily used for image and video data processing. It offers the advantage of preserving spatial/local information by analyzing specific pixels and their surrounding pixels, rather than the entire image.

In contrast, GAN consists of a generator and discriminator. The generator aims to generate data that

closely resembles real data, attempting to deceive the discriminator. On the other hand, the discriminator's role is to distinguish between real and fake data. GAN offers the advantage of generating new images that mimic the target images. This is achieved by training the generator to match the probability distribution of real data. To enable the ML models mentioned earlier to reproduce the SGS component, it is necessary to separate the SGS part and resolved part. Therefore, in this study, a two-dimensional Fast Fourier Transform (FFT) is employed to accomplish this task. The two-dimensional FFT is a mathematical technique used to convert spatial data into frequency domain data. The transformation into the frequency domain facilitates the removal of specific frequency components. By isolating the high-resolution LES SGS part, the focus of this research lies in training the ML models to generate velocity fields that accurately correspond to the SGS part, achieving similar accuracy to that of high-resolution LES.

1.2 Objectives of Research

LES is a turbulence model commonly employed to simulate instantaneous flow behaviors in various types of flows. The LES turbulence model relies on SGS eddy viscosity models, which assume isotropic turbulence in the SGS, in contrast to the resolved scale where the flow velocity and pressure are directly resolved. However, due to the isotropic assumption and dependence on the local grid size, accurately representing certain turbulent flows with strong shear layers becomes challenging for these existing eddy viscosity models. In this context, a new approach is proposed to reproduce such turbulent flows with reduced computational costs while maintaining similar accuracy to high-resolution LES simulations. The primary objective of this study is to develop ML algorithms capable of generating turbulent motion at the sub-grid scale without relying on existing SGS modeling approaches or specific turbulence assumptions.

The highest resolution LES data, obtained in advance, are initially compared to the results of

numerous previous experiments to establish their reliability as ground truth. To separate the LES turbulent flow into the resolved part and the SGS part, FFT is employed. The SGS part of the highest resolution LES data is analyzed, and suitable CNN and GAN structures are applied to reproduce it. During this step, the optimal CNN and GAN structures for predicting SGS are determined by evaluating various combinations of layers, nonlinear functions, and optimizations. Detailed information about the structures of the machine learning algorithms required to generate SGS turbulent flow is provided. This comparison enables the assessment of the algorithms' ability to handle different grid sizes and Reynolds numbers, thus evaluating their scalability.

After successfully generating the velocity, the subsequent step involves extending the analysis to include the generation of turbulent characteristics as well. This study purposes to assess the accuracy of machine learning techniques in simulating turbulence characteristics using time-averaged velocity data and grid points. The input data comprises time-averaged velocities and grid points for a jet flow, while the output data includes time-averaged Turbulent Kinetic Energy (TKE), production, and dissipation. The study initially trains on a Reynolds number of 10,000 and conducts testing on a Reynolds number of 15,000. The normalized root mean square error (NRMSE) and R-squared values are employed to analyze the performance of the ML models in simulating TKE using only time-averaged velocity as input. In order to achieve the main goal of this study, the following three sub-objectives need to be accomplished beforehand: (1) Design and evaluate machine learning algorithms to optimize the generation of the jet flow field. (2) Investigate the applicability of the best-performing machine learning algorithm in generating turbulence characteristics in jets. (3) Compare and assess the computational efficiency of implementing the machine learning algorithm with relying solely on Computational Fluid Dynamics (CFD) calculations.

CHAPTER 2. FUNDAMENTAL BACKGROUDS

2.1 Jets

When a fluid is discharged through an orifice and encounters another fluid, differences in velocity, temperature, or contamination can result in momentum and buoyancy. The type of flow that forms depend on the dominant factor; if momentum is dominant, a jet is formed, while if buoyancy is dominant, a plume is formed. In cases where both factors are present, a buoyant jet or forced plume can form (Abramovich et al., 1984). Among them, jets have been widely used in hydraulic and aerodynamic engineering to address various environmental issues such as the discharge of water waste in domestic, agricultural, and industrial areas, chemical release accidents caused by marine traffic and offshore exploration, and the construction of marine structures and waste disposal and treatment systems (Mossa & Davies, 2017). The use of jet flows is intended to improve turbulent mixing of chemical components, increasing efficiency, and decreasing pollutant concentration. Due to its velocity difference between the centerline and surroundings, the jet flow involves high mass, momentum, and energy exchange, resulting in high mixing efficiency. However, the complex nature of jet flows necessitates studying characteristics such as velocity, Reynolds number, and turbulent kinetic energy.

Fischer et al. (1979) emphasized that jet flow characteristics are determined by three parameters: jet parameters, environmental parameters, and geometrical parameters. Jet parameters refer to the initial conditions of the jet at the inlet, such as the inlet velocity, and the properties of the jet substance, such as its temperature and salinity. Environmental parameters describe the conditions of the surrounding fluid, such as the presence of a current or stratification of the ambient fluid. Geometrical parameters are related to the shape and inclination of the jet inlet. Depending on these various parameters, jet flows can take on different shapes. To better understand turbulent flow, it is necessary to examine the shape of the jet flow under various conditions. This study investigates the characteristics of a jet flow that is discharged vertically upward from the bottom of an ambient fluid, where the jet inlet has a round shape.

Figure 1 illustrates the ideal experimental configuration of jets ejected into an ambient fluid of the same type via a nozzle with diameter D with velocity U_J. The Reynolds number, a non-dimensional parameter, is defined as $Re = U_J d/\nu$. Jets exhibit statistically stationary and axisymmetric flow, meaning the flow characteristics are independent of time and circumferential coordinate, but depend on the axial and radial coordinates. The velocity components in the x, r, and θ directions are denoted by U, V, and W. As the velocity is independent of circumferential coordinate, the centerline velocity is denoted as in terms of the temporally mean axial velocity field $< U(x, r, \theta) >$ as

$$U_0(x) = U(x, 0,0). \tag{1}$$

The jet half width $r_{1/2}(x)$ is defined as

$$U(x, r_{\frac{1}{2}}, 0) = 1/2 U_0(x). \tag{2}$$

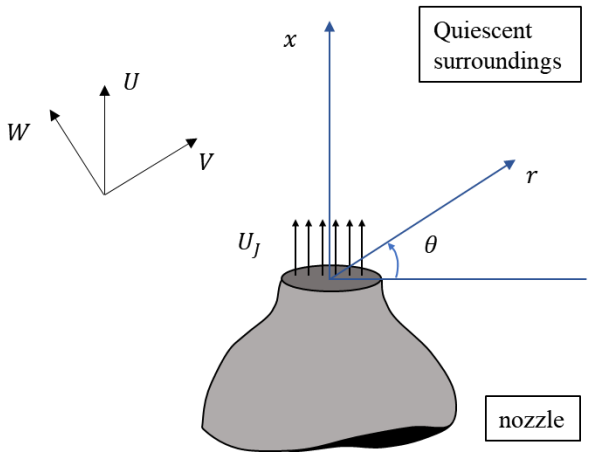

Figure 1. A sketch of a round jet experiment with polar-cylindrical coordinate system, taken from Pope (2000).

Figure 2 depicts the mean axial velocity profiles observed at two points along the jet, showing how the jet decays and spreads with increasing axial distance. It is clear that the jet decay is characterized by a decrease in the centerline velocity $U_0(x)$, while the jet spread is characterized by an increase in the half-width $r_{1/2}$.

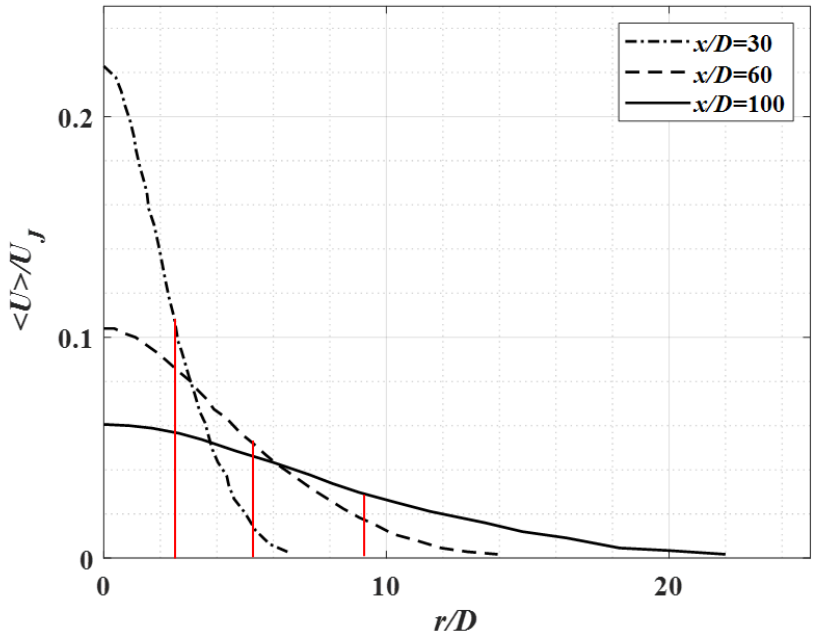

Figure 2. Radial profiles of mean axial velocity in a turbulent round jet at Re 95,500. The red lines indicate the half width of the profile, taken from Hussein et al., (1994).

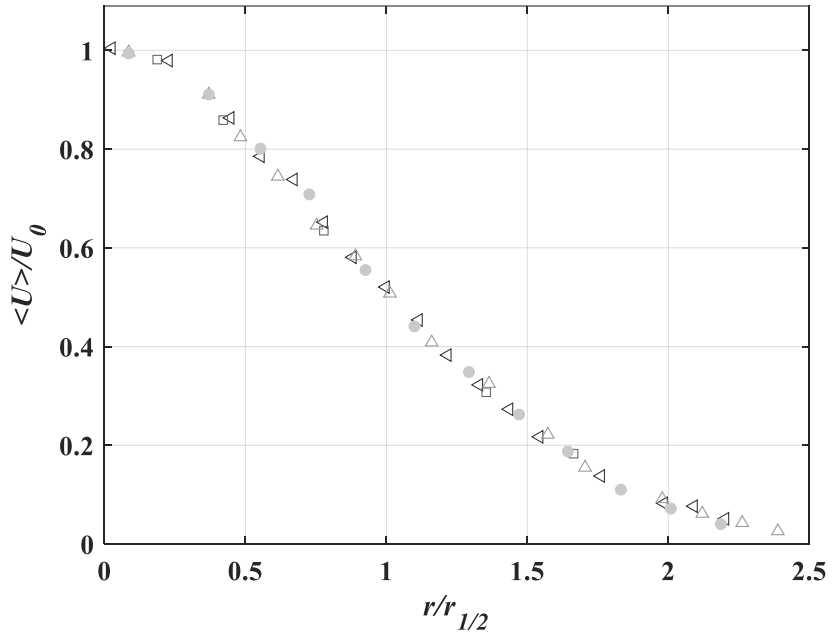

Figure 3. Radial profiles of mean axial velocity in a turbulent round jet, Reynolds number of 100,000. The radial distance is divided by the half width, adapted by Wygnanski and Fiedler (1969).

However, the shape of the mean axial velocity profiles does not change in Figure 3 , where the profiles plotted against $r/r_{1/2}$ collapse onto a single curve.

Self-similarity is an important factor that arises in fully turbulent flows. It refers to the property that certain aspects of the flow, such as the shape of the velocity profile, remain unchanged as the flow evolves in the downstream direction shown in Figure 3. In terms of the velocity profile, this means that the profile at one axial location can be expressed to the profile at a different axial location by a simple scaling factor. To quantify this property, characteristic scales $Q_0(x)$ and $\delta(x)$ are defined for dependent variable Q and the independent variable y as functions of x. Then, scaled variables are defined by (Pope, 2000)

$$\xi \equiv \frac{y}{\delta(x)} \tag{3}$$

$$\tilde{Q}(\xi, x) \equiv \frac{Q(x,y)}{Q_0(x)}. \tag{4}$$

If two variables, $\tilde{Q}(\xi, x)$ and $\tilde{Q}(\xi)$, are the same, it means that the variable Q is self-similar with respect to the scaling variable ξ. It is characterized by a self-similar curve that closely resembles a Gaussian distribution. This can be mathematically expressed as:

$$U(x,r) = U_0(x)\exp\left(-\frac{r^2}{2\sigma^2}\right). \tag{5}$$

Here, $\sigma(x)$ represents the standard deviation of the velocity profile from the centerline.

Returning to velocity profile, Figure 4 shows the inverse of $U_0(x)$ plotted against x/D. The experimental data are observed to lie on a straight line, which can be represented by the equation:

$$\frac{U_0(x)}{U_J} = \frac{B}{(x-x_0)/D}. \tag{6}$$

Here, x_0 and B are defined as the virtual origin and empirical constant, respectively. It is found that the jet spreads linearly, and the spreading rate is defined as:

$$S \equiv \frac{dr_{1/2}(x)}{dx}. \tag{7}$$

The spreading rate is also constant and can be rewritten as

$$r_{1/2}(x) = S(x - x_0) \tag{8}$$

for x in self-similar region.

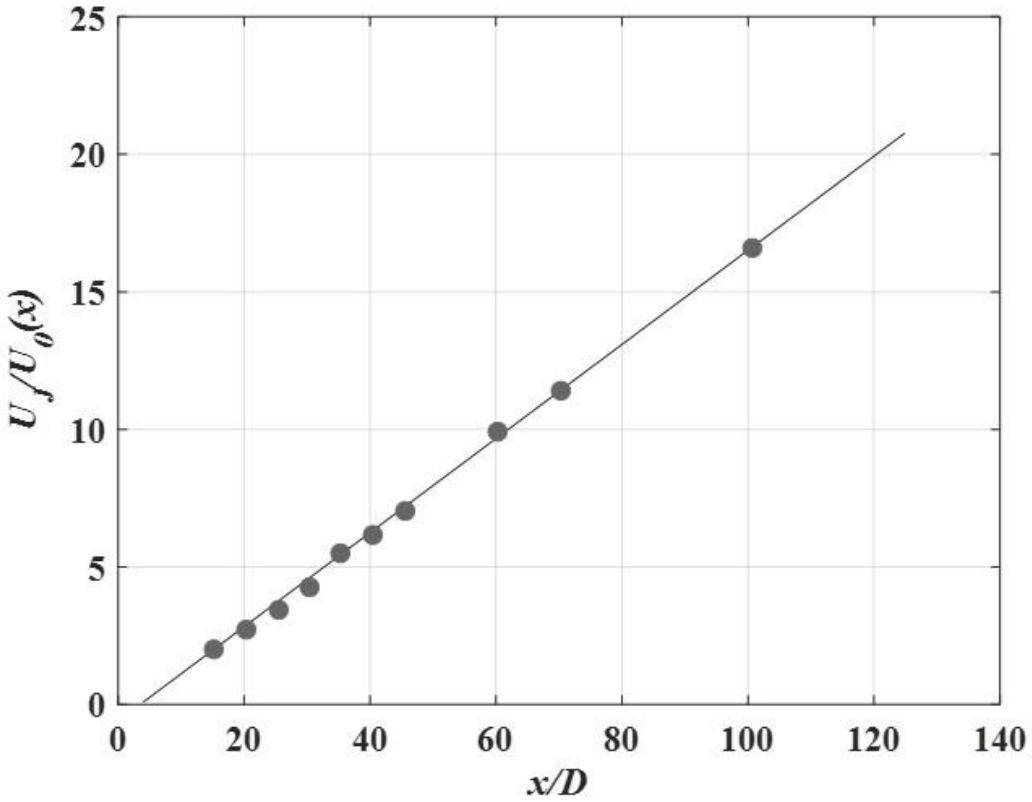

Figure 4. The inverse of the mean centerline velocity along the centerline in a turbulent round jet at Reynolds number of 95,500; measurements of Wygnanski and Fiedler (1969).

Table 1. The spreading rate S and velocity decay constant B for turbulent round jets (Pope, 2000).

	Panchapakesan and Lumley (1993)	Hussein et al. (1994), hot-wire data	Hussein et al. (1994), laser-Doppler data
Re	11,000	95,500	95,500
S	0.096	0.102	0.094
B	6.06	5.9	5.8

Figure 5 depicts a simple round jet that was studied in this research. The flow of the simple round jet can be divided into two zones: the zone of flow establishment (ZFE) and the established flow zone (ZEF). The criterion that distinguishes between the two regions is the ratio of the size of the jet inlet to the distance from the inlet, x. The criterion for distinguishing between the two regions is $x/D < 6.2$, and the flow velocity profiles in these two zones are distinct (List, 1982). The velocity profiles are as follows:

1. In the zone of flow establishment (ZFE)

$$U = U_J, c = c_J; \qquad r \le R \tag{9}$$

$$U = U_J exp\,[-(r-R)^2/b^2\,], c = c_J exp\,[-(r-R)^2/\lambda^2 b^2]\,; \qquad r > R \tag{10}$$

2. In the zone of established flow

$$U = U_J \exp\,[-\left(\frac{r}{b}\right)^2], \; c = c_J \exp\,[-\left(\frac{r}{\lambda b}\right)^2] \tag{11}$$

c, c_J represent a concentration and jet inlet concentration, respectively, while $b(x)$ is width of jet. λ is a constant that was determined through laboratory experiments.

Figure 5 shows that the ZFE can be further subdivided into the potential core and the mixing zone based on the distance from the center of the inlet. Due to the momentum exchange between the core

and the quiescent fluid, the core's width decreases as it moves away from the inlet. Conversely, the width of the mixing layer increases with distance from the inlet. Specifically, the entrainment process and the spreading rate of a turbulent jet are determined by the large and dominant eddies that extend across the entire width of the jet (Lee et al., 2003). Based on these jet flow properties, this study focuses on reproducing the small eddies rather than the large eddies.

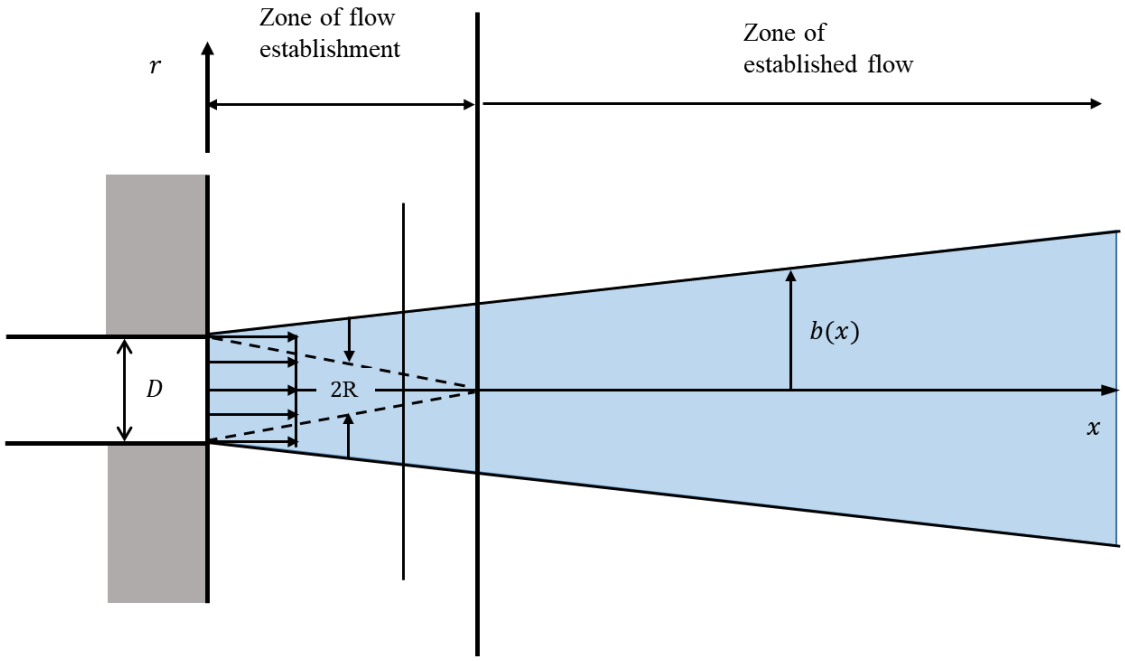

Figure 5. Simple jet showing zone of flow establishment and established flow (Lee et al., 2003)

If the streamwise and radial coordinates are denoted by x and r, and the velocities are u and v, the continuity and x -momentum equations are as follows (List, 1982):

$$\rho\frac{\partial u}{\partial x}+\rho\frac{1}{r}\frac{\partial}{\partial r}(rv)=0 \tag{12}$$

$$\rho u\frac{\partial u}{\partial x}+\rho v\frac{\partial u}{\partial r}=-\frac{1}{r}\frac{\partial}{\partial r}(r\overline{\rho u'v'}). \tag{13}$$

The x-momentum equation can be rewritten as:

$$\rho u\frac{\partial u}{\partial x}+\rho v\frac{\partial u}{\partial r}=\rho u\frac{\partial u}{\partial x}+\rho v\frac{\partial u}{\partial r}+\rho u\left(\frac{\partial u}{\partial x}+\rho\frac{1}{r}\frac{\partial}{\partial r}(rv)\right)=-\frac{1}{r}\frac{\partial}{\partial r}(r\overline{\rho u'v'}). \quad (14)$$

Above equation is then multiplied by $2\pi r$ and integrated from $r=0$ to infinity, yielding:

$$\frac{d}{dx}\int_0^\infty \rho u^2 2\pi r dr=0. \quad (15)$$

Substituting the assumed velocity profile, the equation can be modified as:

$$M(x)=\int_0^\infty u^2 2\pi r dr=\frac{\pi}{2}(U_0^2 b^2)=U_m^2\frac{\pi D^2}{4}. \quad (16)$$

Also, the tracer mass conservation is defined:

$$u\frac{\partial c}{\partial x}+v\frac{\partial c}{\partial r}=-\frac{1}{r}\frac{\partial}{\partial r}(r\overline{v'c'}). \quad (17)$$

Similar to momentum equation, the tracer mass flux can be written as:

$$\frac{d}{dx}\int_0^\infty \rho c 2\pi r dr=0 \quad (18)$$

$$\int_0^\infty uc 2\pi r dr=\frac{\pi\lambda^2}{1+\lambda^2}(U_0 c_0 b^2)=U_m c_m\frac{\pi D^2}{4}. \quad (19)$$

The various round jet equations are summarized in Table 2.

Table 2. Round jet formulation for zone of established flow ($x \geq 6.2D$) with jet diameter D, source velocity U_J, and momentum flux M_0 (List, 1982)

Jet width	$b = 0.114x$	
Centerline velocity	$\frac{U_0}{U_J} = 6.2\left(\frac{x}{D}\right)^{-1}$	$U_0 = 7.0M_0^{1/2}x^{-1}$
Centerline concentration	$\frac{c_0}{c_J} = 5.26/\left(\frac{x}{D}\right)^{-1}$	
Centerline dilution	$Dl = 0.19x/D$	
Average dilution ratio	$\overline{Dl} = 0.32x/D$	$Dlr = 0.286M_0^{1/2}x$

2.2 Machine Learning

Figure 6 shows the relationships between artificial intelligence, machine learning, deep learning, supervised learning, and unsupervised learning. Artificial intelligence (AI) is a technology that embodies human-like abilities, such as learning, reasoning, perceptual, and language comprehension abilities in a computer (Ongsulee, 2017). Therefore, it has the potential to be applied in various areas, such as speech comprehension, game playing, driving a car, and interpreting complex data (Pannu, 2015). However, it requires explicit programming by humans to solve the issues. On the other hand, machine learning, a subfield of AI, has an ability to automatically learn and program rules based large amount of data (Helm et al., 2020). Through sufficient training and correction, machine learning algorithms automatically identify an appropriate model that can produce accurate outputs even when presented with previously unseen data. Another definition of ML is that it involves algorithms that can learn patterns in high-dimensional spaces without explicit programming or direction from humans (Prado, 2018). It differs from traditional statistical modeling in that it trains the rules instead of solely predicting outcomes. ML involves several steps in learning these rules, including image analysis, object detection, and object recognition.

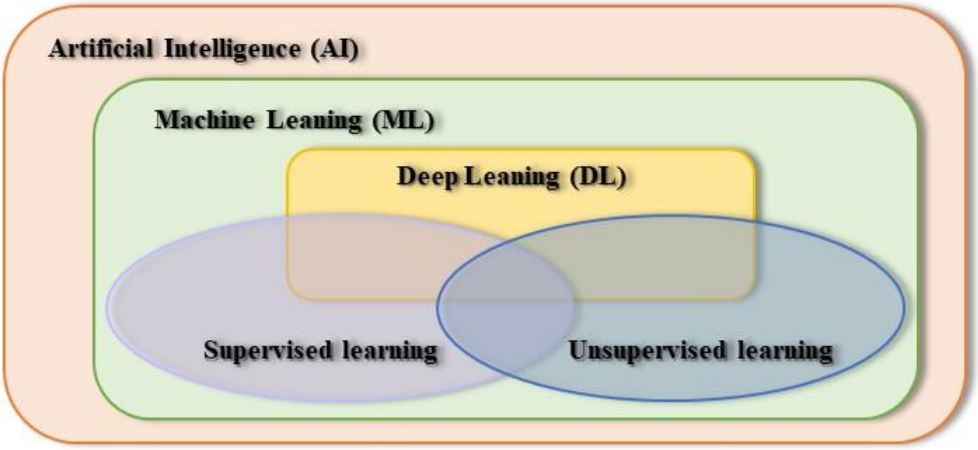

Figure 6. Relationships among artificial intelligence, machine learning, deep learning, supervised learning and unsupervised learning.

ML has several main categories related to learning approaches: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is a training process using labeled data. In other words, the algorithms receive a set of input data and the output of each input data. It is frequently used in classification and regression tasks. Classification is a process of dividing a large dataset into distinct groups or classes based on specific features or attributes. In contrast, regression involves predicting a continuous value of a target variable based on inputs. Unsupervised learning algorithms, on the other hand, use data without any labels, and determine the correct answers without any expected output. This algorithm has a high degree of freedom and is applied when solving clustering problems, where the goal is to group data into meaningful clusters based on their similarities. Another important application of unsupervised learning is dimensionality reduction, which involves simplifying complex data by identifying and removing redundant or irrelevant features, thus improving the performance of subsequent analysis tasks. Reinforcement learning has no expected output and is trained using trial and error. In other words, it rewards and acts in the desired direction when the desired output is achieved.

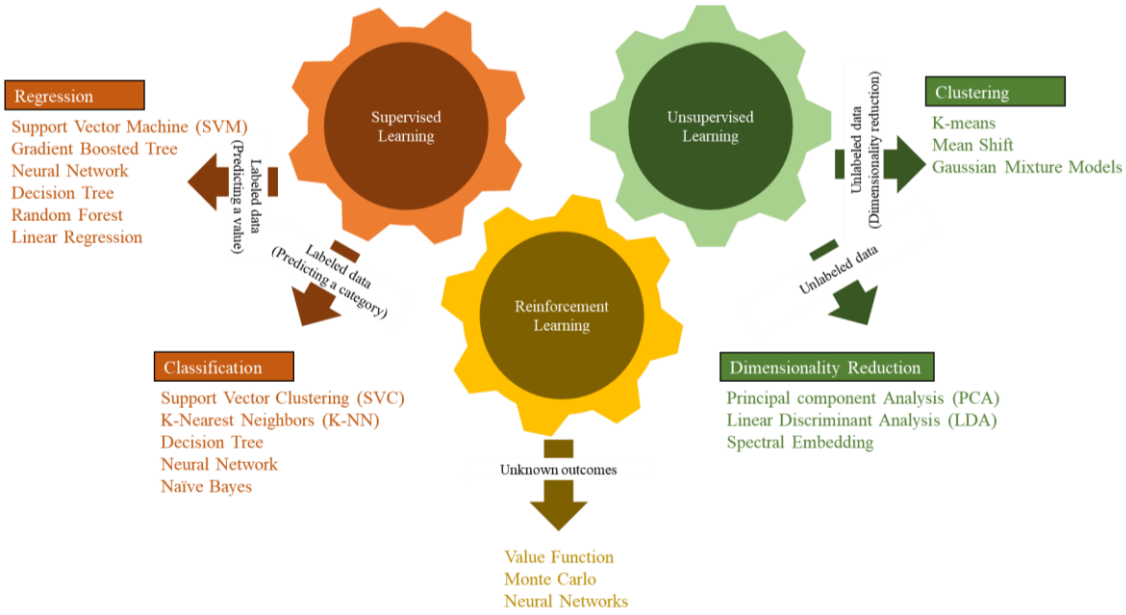

Figure 7. Machine learning types and applications

Deep learning (DL) has been developed to solve the multi-step problem of ML. DL, also known as deep hierarchical learning or deep machine learning, is a subfield of ML. DL can be trained in single step, but the explanation of the results and causes is insufficient. DL is based on an artificial neural network (ANN), which is an algorithm that mimics the structure and operation of the human neural network (Vemuri, 1988). ANN is trained through the connection of artificial neurons, and in this process, it finds appropriate weights and bias values. In other words, DL consists of these neuron layers, and the word 'deep' simply refers to the concept of using numerous layers.

2.2.1 Basic of Deep Learning

Now, the fundamentals of ANN are discussed briefly. As mentioned previously, ANN is a deep learning algorithm that refers to the human neural network, so it consists of neurons, which are the smallest performance components (Rahman et al., 2019). These neurons form a layer, and these layers

constitute a typical ANN algorithm. Typically, there are usually three or more layers, and they are called input layer, hidden layer, and output layer. The first layer of the algorithm is referred to as an input layer, and as its name implies, it is responsible for receiving input data. Similarly, the output layer is the final layer and is responsible for rearranging and exporting the output data to a desired size. The hidden layers determine how much the previous data should be weighted up or down before being passed to the other layers (O'Shea & Nason, 2015). Each layer is connected by coefficients (weights), bias, and non-linear functions (Zupan, 1994).

In Figure 8a, there is a 4 layered ANN algorithm, and the layers have connections each other. In these connections, each neuron receives input data, multiplies weights, adds bias, and then passes through a non-linear function to generate output that the following neuron receives (Agatonovic-Kustrin & Beresford, 2000). Specifically, input data manipulation operations are separated into linear and non-linear components. Weight is one of the linear components that is multiplied with the input value. Typically, initial value of the weight is determined randomly, and this value is modified during algorithm training. A significant input value is multiplied by a weight with a large absolute value, and a value close to 0 is multiplied in the opposite case. In addition to weights, bias is also a linear component. It is added to the output of multiplying the input by a weight, essentially to alter the input's range. Finally, the activation function is applied to transform this linear relationship into a nonlinear one.

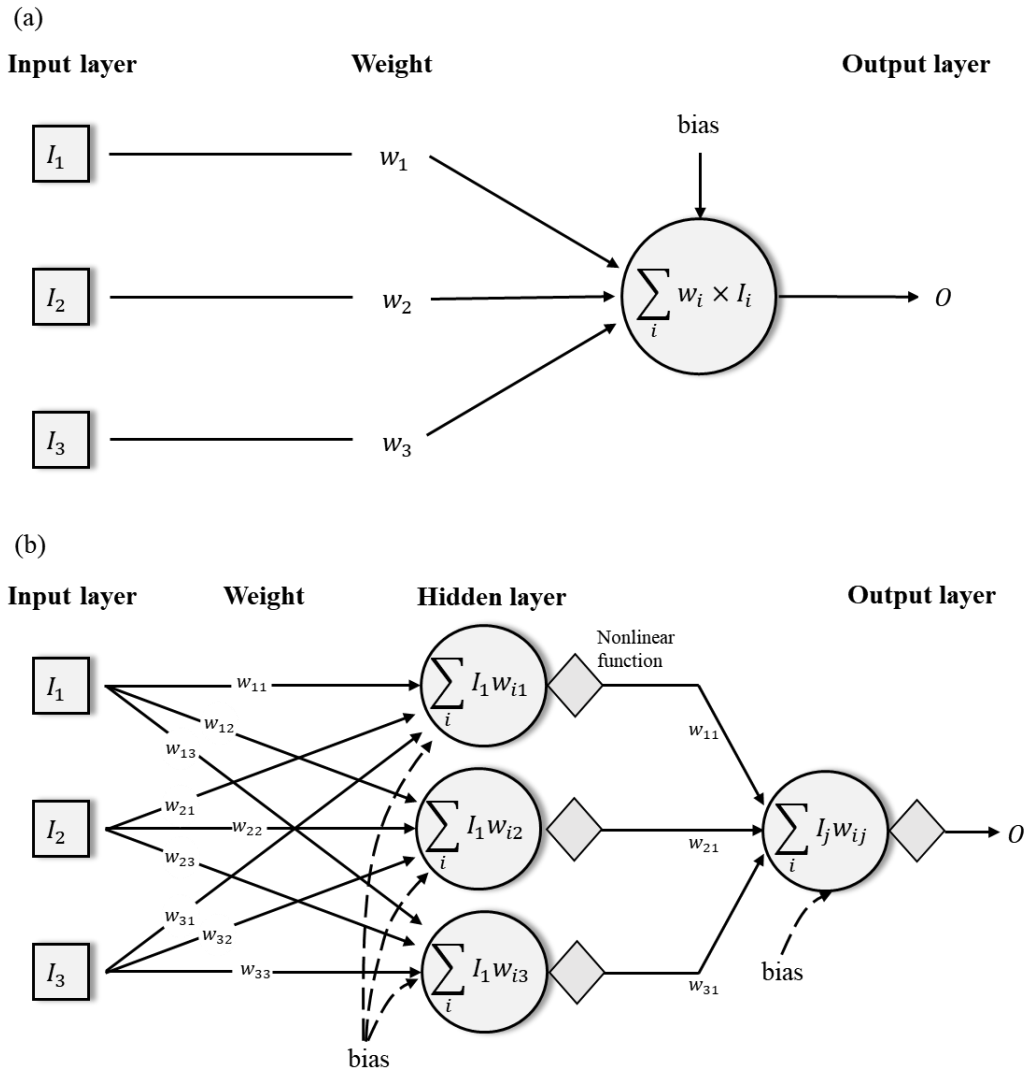

Figure 8. (a) Typical architecture of artificial neural network consisting of input, hidden and output layer and (b) artificial neuron example with 3 input neurons and activation function, ReLU.

Figure 8b is an enlarged version of the layer connections, as shown in Figure 8a. Three neurons in the input layer are connected to a single neuron in the next layer by multiplying their weights, adding a bias, and using a nonlinear activation function, Rectified Linear Unit (ReLU). In this way, the movement of the input value through the hidden layer to the output layer is called as forward propagation. As its name suggests, it goes only one direction, forward-pass. However, when moving only forward, weight

or bias value cannot be updated. Thus, the backpropagation is required to update the weight and bias to make better outcomes. The back propagation uses a loss (cost) function to calculate the error value, and then updates the weight and bias values in the direction which the error decreases.

Sigmoid, Rectified Linear Unit (ReLU), softmax, tanh, and leaky ReLU are the most frequently employed activation functions. These activation functions are summarized below, and Figure 9 depicts the activations.

• Sigmoid: The input data is valid for all real numbers, while the output data is in the interval [0, 1]. The S-shaped curve of the Sigmoid nonlinear function makes it continuously differentiable. It is represented mathematically:

$$f_{sigmoid}(x) = \frac{1}{1+e^{-x}}. \quad (20)$$

• ReLU: It is one of common non-linear functions in CNN algorithm (Alzubaidi et al., 2021). It converts all numbers to positive numbers or 0. When the output is zero, the corresponding neuron deactivates. It indicates that ReLU has a low computational cost.

$$f_{ReLU}(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \quad (21)$$

• Softmax: It combines multiple sigmoid functions. While Sigmoid is used for binary classification, Softamx can be employed for multi-classification (Sharma et al., 2017). Its benefit is returning the probability for each class. It can be expressed as

$$f_{softmax}(x) = \frac{e^{-x}}{\sum e^{-x_i}} \ for group \ i. \quad (22)$$

• Tanh: appears S-shaped and symmetrical about the origin. Hence, it is continuous and differentiable. Also, it has steep gradient. The output ranges from -1 and 1.

$$f_{tanh}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (23)$$

• Leaky ReLU: Positive numbers come out directly, whereas negative numbers come out by down-scaling. The down scaling number m is usually small.

$$f_{LeakyReLU}(x) = \begin{cases} x & x > 0 \\ mx & x \leq 0 \end{cases} \tag{24}$$

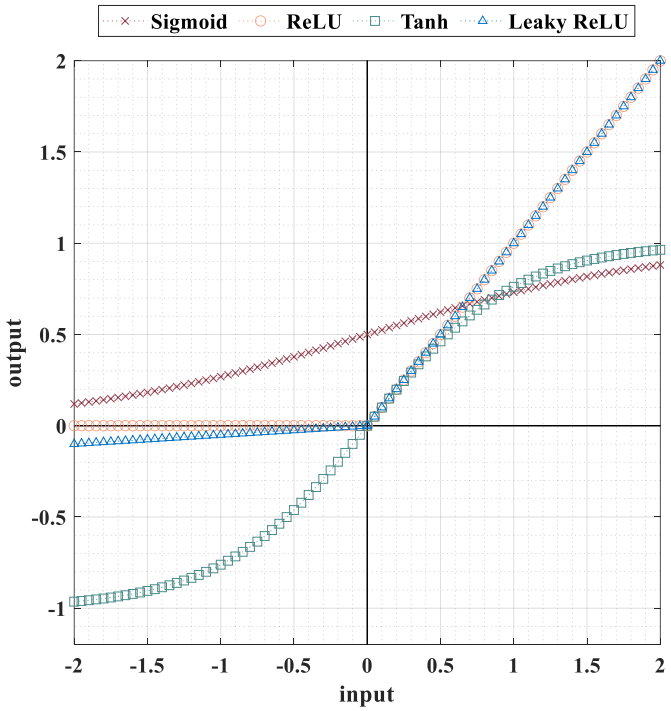

Figure 9. Non-linear activation functions: Sigmoid, Rectified Linear Unit (ReLU), Tanh, Leaky ReLU. Whereas Sigmoid and Tanh have a range of outputs, ReLU and Leaky ReLU are not constrained in the positive direction. In Leaky ReLU, m is 0.05.

The loss function can be divided into two groups: classification and regression, according to the objectives of deep learning algorithms. Various loss functions are summarized in Table 3.

Table 3. Loss function for classification and regression

classification	
0-1 loss function	$L(y,\dot{y}) = \begin{cases} 1 \; if \; y\dot{y} < 0 \\ 0 \; if \; y\dot{y} \geq 0 \end{cases}$
Perceptron loss function	$L(y,\dot{y}) = \begin{cases} -y\dot{y} \; if \; y\dot{y} < 0 \\ 0 \quad if \; y\dot{y} \geq 0 \end{cases}$
Cross entropy loss	$L(y,\dot{p}) = -log\dot{p}$ where $\dot{p} = \begin{cases} p \quad if \; y = 1 \\ 1-p \; if \; y \neq 1 \end{cases}$
Sigmoid entropy loss	$L(y,\dot{p}) = -log\dot{p}$ where $\dot{p} = \begin{cases} p \quad if \; y = 1 \\ 1-p \; if \; y \neq 1 \end{cases}$
Softmax cross entropy loss	$L\big(y, p(y\|x)\big) = -p(y\|x)$
Exponential loss	$L(y,\dot{y}) = e^{-y\dot{y}}$
Regression	
Square loss	$L(y,\dot{y}) = (y-\dot{y})^2$
Absolute loss	$L(y,\dot{y}) = \|y-\dot{y}\|$
Huber loss	$L(y,\dot{y}) = \begin{cases} (y-\dot{y})^2/2 \quad if \; \|y-\dot{y}\| \leq \delta \\ \delta\|y-\dot{y}\| - \delta^2/2 \quad otherwise \end{cases}$

x: input data, y: real value, $\dot{y}$: generated value, L: loss function
p and $\dot{p}$: probability δ: parameter

There are four representative loss functions for classification. 0-1 loss function has only two loss values, 0 and 1. The loss is 1 if the real value and generated value have same sign (classification to the same group), otherwise the loss is 0, shown in Figure 10a. It sets the loss to 1 regardless of the difference between the real and generated values. Consequently, it has the disadvantage that it is difficult to determine how much to modify, so it is rarely used. Therefore, the perceptron loss function is developed to assess the difference. The loss is 0 when the real and generated value have same sign, otherwise loss is absolute value of generated value (Figure 10b). These two loss functions are simple to use, but they are not robust for noise data (Wang et al., 2022).

Cross entropy loss is another model commonly used to measure deep learning models. It has a singularity that probability is an output. So, it helps the generated distribution to have similar real data distribution. For example, the loss function predicted a 10%, 65%, and 25% probability, respectively, that the outcome would be classified into groups 1, 2, and 3. And if the actual outcome belongs to group 2, the cross-entropy loss is $-(0 \times \log(0.1) + 1 \times \log(0.65) + 0 \times \log(0.25))$. Thus, the cross-entropy loss function can also be used for multiclass classification and is also known as the log loss function (Wang et al., 2022). If it is combined with softmax or sigmoid activation, it is called as softmax or sigmoid cross entropy loss. Exponential loss is also another loss, and it is superior to the 0-1 loss function. Because the loss is computed by applying an exponent to the product of the generated and actual values (Figure 10c). This loss function offers two benefits (Wang et al., 2022). Firstly, it can be continuous and differential. Secondly, it can impose a penalty if the true and generated values have different sign. To be specific, the error gradient become sharper, as the loss is large. So, when the generated value is incorrectly labeled, this loss function helps the model to be converge easily.

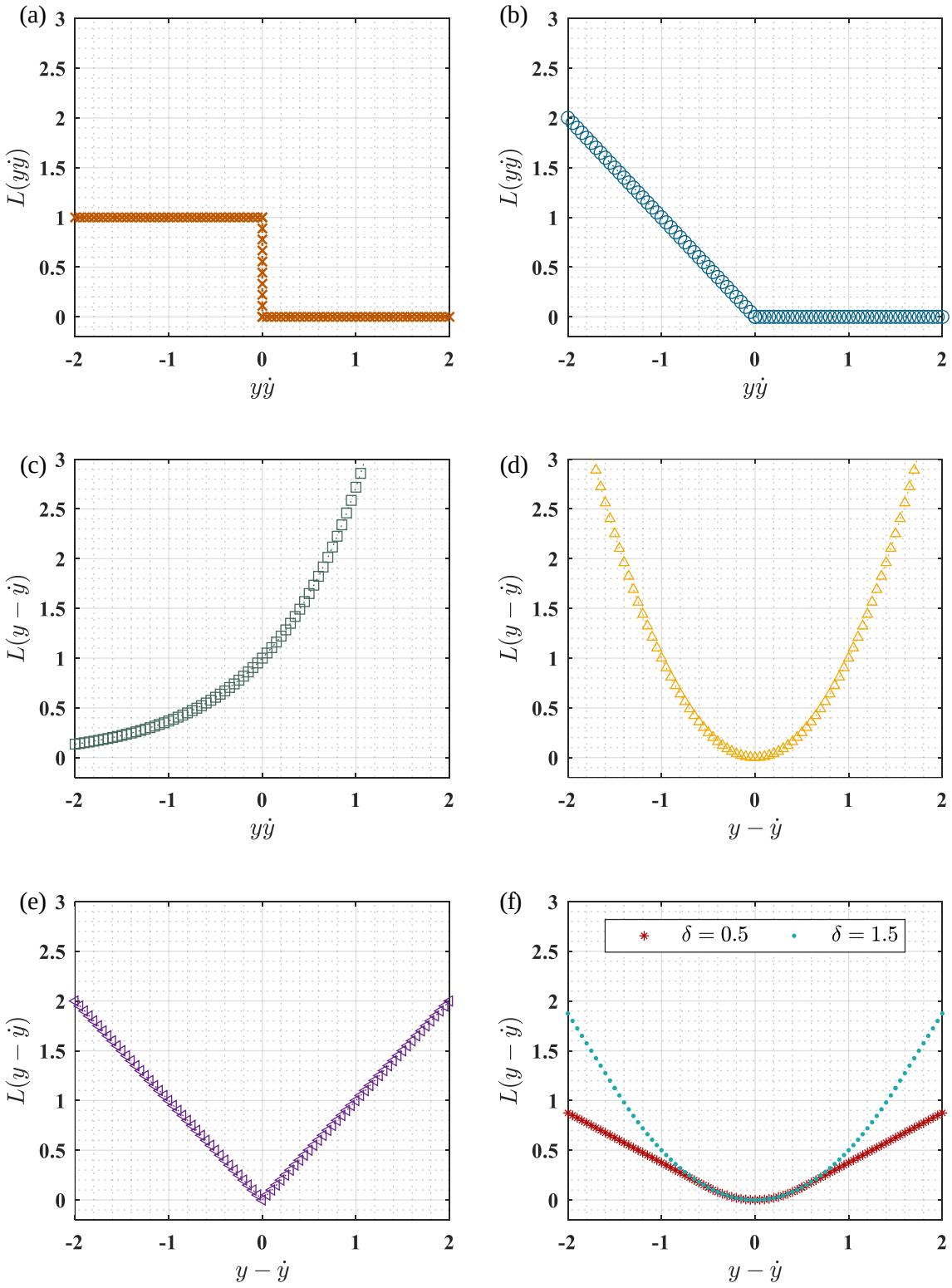

Figure 10. Loss function for classification and regression; (a) 0-1 loss function, (b) perceptron loss function, (c) exponential loss, (d)square loss, (e) absolute loss and (f) huber loss

For regression problems, there are several loss functions, including the square loss function, the absolute loss function, and the huber loss function (Sharma, 2017). The square loss function calculates the square of the difference between the real and generated values. If the difference is large, so is the gradient (Figure 10d). While, as the loss value approaches 0, the gradient becomes smaller. It has an advantage for rapid convergence and precision (Wang et al., 2022). Absolute loss measures a difference between the real value and the generated value shown in Figure 10e. It is robust when outliers are trained because the loss does not increase sharply even if the difference is significant. When the generated value is close to the actual value, however, it has a larger loss than the square loss function, so it is less popular than the square loss function. Huber loss is a loss function combined with square and absolute loss functions. This loss function takes one parameter (δ) and compares its magnitude to the loss value to determine the appropriate loss equation. If the difference between the real and generated values is less than the parameter, the squared loss is employed; otherwise, the absolute loss is employed (Figure 10f). Consequently, it is possible to state that this model possesses both the benefits of an absolute loss function and a square loss function (Wang et al., 2022). First, when training the outlier, the absolute loss function can be used to prevent the loss from rising sharply. When it is very close to the real value, on the other hand, it can be learned by gradually adjusting the weights and bias using the square loss function. However, setting the parameter to proper value is difficult.

When the loss value is obtained by the loss function introduced above, the weight and bias are adjusted accordingly, which is called back propagation. This method is founded on the theory of gradient descents (GD). If the loss and weight at epoch, i, are set to $L(w_i)$ and w_i, respectively, then the weight at next epoch using GD can be defined as:

$$w_{i+1} = w_i - \gamma \nabla L(w_i). \tag{25}$$

As shown by the equation, the weight changes in the opposite direction of the loss function's slope. If the loss function has a positive slope, the weights must be changed to a value that causes the loss

function to shift in the opposite direction. The rate at which the weights change over one epoch is known as the learning rate. If the learning rate is too high, the optimal value is bypassed, whereas if it is too low, a local minimum may be found (Goodfellow et al., 2014). After all train data is learned, GD modifies the weights and biases, so it takes a long time to find the optimal weight and bias. To solve this problem, the stochastic gradient descent (SGD) algorithm was developed. After the batch has been trained, the weight is adjusted. The batch is a subset of the entire data set. This method finds optimal values more quickly than gradient descent, but it still does not solve the disadvantage that the learning rate has a large effect. (Wang et al., 2022).

When using gradient descent, it is important to note that weights or biases can reach local minimums or saddle points. When the weights are updated, the learning gradient, which determines in which direction to train, is a crucial property for solving this problem. Momentum optimizer is the optimizer that the GD optimizer has been improved in terms of learning gradient. When updating the weight, this method reflects the direction in which the weight was previously altered. It is defined as:

$$v_{i+1} = \rho v_i + \nabla L(w_i) \tag{26}$$

$$w_{i+1} = w_i - \gamma v_{i+1} \tag{27}$$

where v_i is the previously weight changed direction, also known as weight increment. ρ is momentum factor value, which ranges from 0 to 1 (Wang et al., 2022). If the momentum factor is low, it cannot avoid local minimum or maximum. While, when the momentum factor is too high, it is easy to skip the global minimum or maximum.

Optimizers are concerned not only with the gradient but also with the learning rate. Adagrad is an optimization that enhances the learning rate of gradient descent. It has a variable learning rate depending on the frequency of adjustment. It has a high learning rate for infrequent updates, as variables that change infrequently are likely to require significant adjustments to reach the optimal solution. However, it has low learning rate for frequent updates (Ruder, 2016). Dean et al. (2012) told that Adagrad is

more resilient than stochastic gradient descent in large neural network models. Adagrad is defined as:

$$S_{i+1} = S_i + (\nabla L(w_i))^2 \tag{28}$$

$$w_{i+1} = w_i - \frac{\gamma}{\sqrt{S_{i+1}+\varepsilon}} \cdot \nabla L(w_i) \tag{29}$$

In this equation, S_i is accumulating and growing as training continues. Thus, the step size decreases as the algorithm is trained. It indicates that the model can converge as the local point is approached as the step size becomes smaller. Therefore, it is advantageous for convex functions, but difficult to avoid the saddle point (Goodfellow et al., 2014). In order to prevent the step size from becoming too small, RMSprop is an optimization model that is slightly improved by Adagrad. RMSprop sets decay rate (λ) to gradually decrease the accumulated value.

$$S_{i+1} = \chi \cdot S_i + (1-\chi)(\nabla L(w_i))^2 \tag{30}$$

$$w_{i+1} = w_i - \frac{\gamma}{\sqrt{S_{i+1}+\varepsilon}} \cdot \nabla L(w_i) \tag{31}$$

Adam is an optimizer that combines Momentum and RMSprop. Consequently, both gradient direction and learning rate can be considered. It has advantages that are computationally inexpensive and simple to use (Wang et al., 2022). And it needs less tuning than every other optimization method (Goodfellow et al., 2014). However, this model heavily focuses on reducing the computational time, so it may not converge to the optimal solution (Tran, 2019). In addition, it has a decaying problem, so Loshchilov & Hutter (2018) propose the AdamW optimization for addressing Adam's decaying problem. It is mathematically expressed as:

$$v_{i+1} = \chi \cdot v_i + (1-\chi_1)\nabla L(w_i) \tag{32}$$

$$S_{i+1} = \chi_2 S_i + (1-\chi_2) \cdot (\nabla L(w_i))^2 \tag{33}$$

$$\widehat{v_{i+1}} = \frac{v_{i+1}}{1-\chi_1^2},\ \widehat{S_{i+1}} = \frac{S_{i+1}}{1-\chi_2^2} \tag{34}$$

$$w_{i+1} = w_i - \frac{\gamma}{\sqrt{\widehat{S_{i+1}} + \varepsilon}} \widehat{v_{i+1}} \tag{35}$$

where χ_1 and χ_2 represent constants.

The types and characteristics of optimizers are summarized in Figure 11a. It shows that momentum takes into account gradient direction, whereas Adagrad considers the learning rate. Moreover, Adam considers both the gradient direction and learning rate. Figure 11b shows how each optimization algorithm traverses the contour of the loss surface (Ruder, 2016). Due to the fact that SGD is an algorithm that has not improved either the learning direction or the learning step, it is evident that the learning speed is slow. Consequently, it is the sole optimization algorithm that has not yet reached its lowest point. Momentum and NAG are optimizers focused on the learning direction, so they have shifted various paths on loss contour. However, the routes of Adgrad, Adadelta, and RMSprop's routes are relatively straightforward because they primarily consider the learning rate. This phenomenon is also shown in Figure 11c (Ruder, 2016). If the path can be known from the Figure 11b, then the training speed of each optimization can be verified in Figure 11c. SGD cannot identify the saddle point, so it does not move away from it. On the other hand, all other algorithms are moving toward the global minimum because these algorithms recognize that they are in a saddle point. The learning rate improved optimizer, Adagrad, Adadelta, and RMSprop, move fast than the other model. As the gradient along the path from the saddle point to the global minimum increases, the learning step increases, and as a result, the training speed of these algorithms increases. In other words, it is evident that the computational cost of these models is low. Due to the slow learning rate, Momentum and NAG, in which only the gradient is improved, move slowly. Therefore, each optimizer has its own strengths and weaknesses, and it is essential to select an optimizer based on the characteristics of the training data.

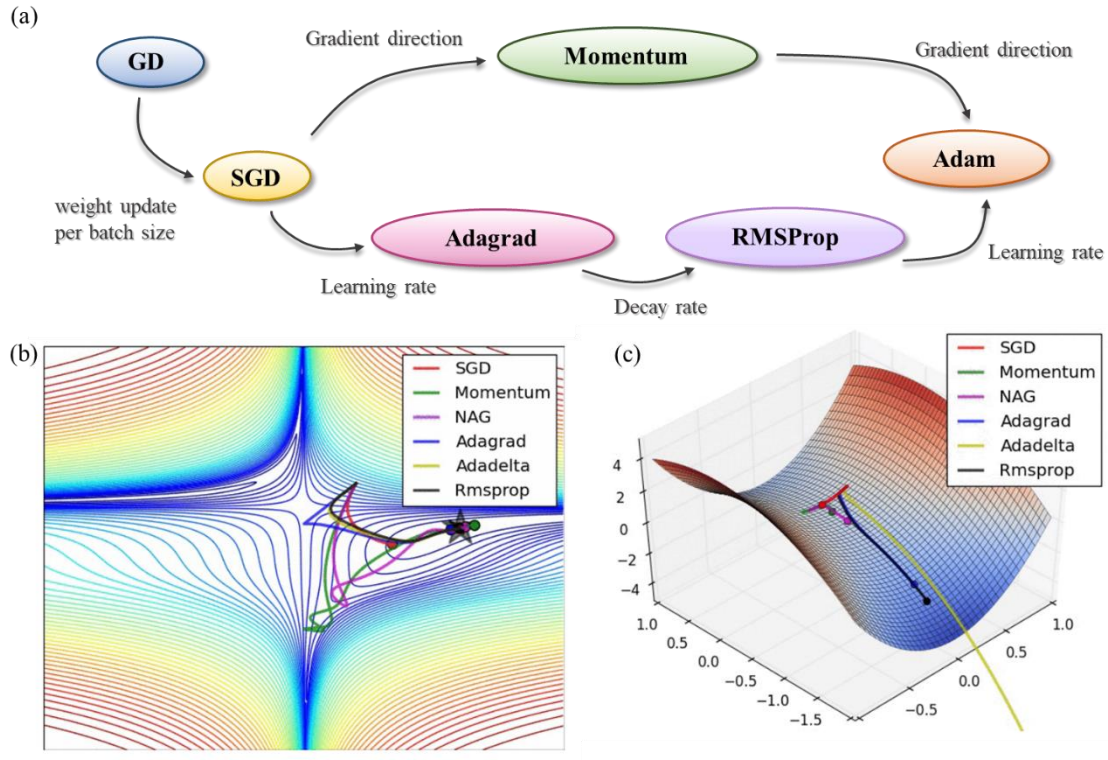

Figure 11. (a) Optimizer types and characteristics, (b) training paths of optimizer on loss contour (Ruder, 2016), and (c) training speed of optimizer on saddle point (Ruder, 2016)

A single execution of forward-pass and back propagation over the entire data set is known as an epoch. However, it is difficult to determine the weights and biases that account for the entire data set at once due to the volume of data. So, after dividing the data by batch size, the calculation for each batch is performed. It is known as one iteration, and at this point the weight and bias values are updated. For example, if the total data set of 1000 is divided into 100 data per one batch, then there are ten iterations per one epoch. If 10 epochs are calculated, the weights and bias are updated through 100 iterations. It is possible to avoid under-fitting and over-fitting by setting the algorithm's epoch values appropriately. Figure 12 shows the concept of under-fitting and over-fitting. Underfitting is too simple to even accommodate the trainset (Figure 7a). The condition of overfitting occurs when the training set is over-trained. Consequently, the model performs poorly on data other than the training data (Figure 12c). If the model is trained with the appropriate number of epochs, as illustrated in Figure 12b, we can find

the optimal model.

In the model training process, data sets can be categorized as train, validation, and test sets. Train set are used to tune the ML model, and parameters such as weight and bias are adjusted to fit the training set. Validation set is used only to evaluate a trained model already been trained. In other words, it is not utilized to estimate weights or biases. The error of train and validation set determines the hyperparameters such as epoch. The test set is used to evaluate the model after training is complete, and it is comprised of unbiased data samples.

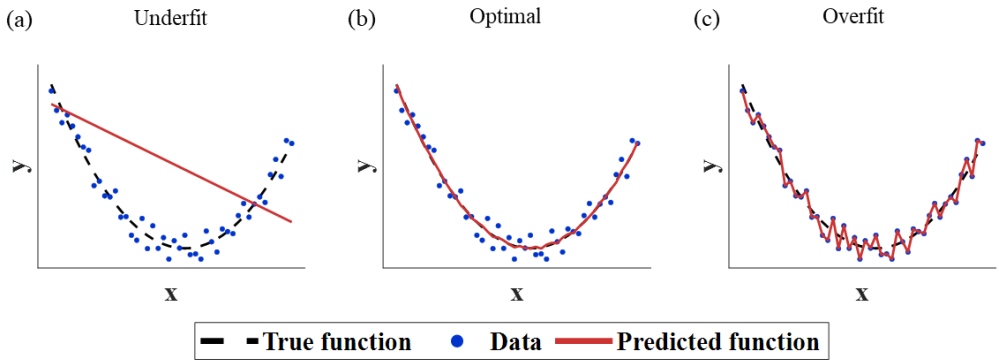

Figure 12. Concept of (a) underfit, (b) optimal, and (c) overfit

2.2.2 Convolutional Neural Network

CNN is a network that, like ANN, receives an input and performs an operation, but the input of each layer in a CNN model is organized in three dimensions (Alzubaidi et al., 2021). The input data consists of height, width, and depth, with height and width being equal. The depth is also referred to as the number of channels. These input data are extracted by the kernel (filter) with sharing the weights. The kernel traverses the input data at a predetermined interval, performs the dot product with the subregion of input data, and obtains the output as the matrix of dot products (Goodfellow et al., 2016). Therefore, the kernel extracts distinct image features from one another, and the resulting image is known as a

feature map. Figure 13 shows a CNN example comprised of an input image, three kernels, and three feature maps. Each kernel extracts unique characteristics, including edge and curve, etc. Here, the width and height of the image match the height of the width of the three-dimensional output, and the depth is three because there are three feature maps.

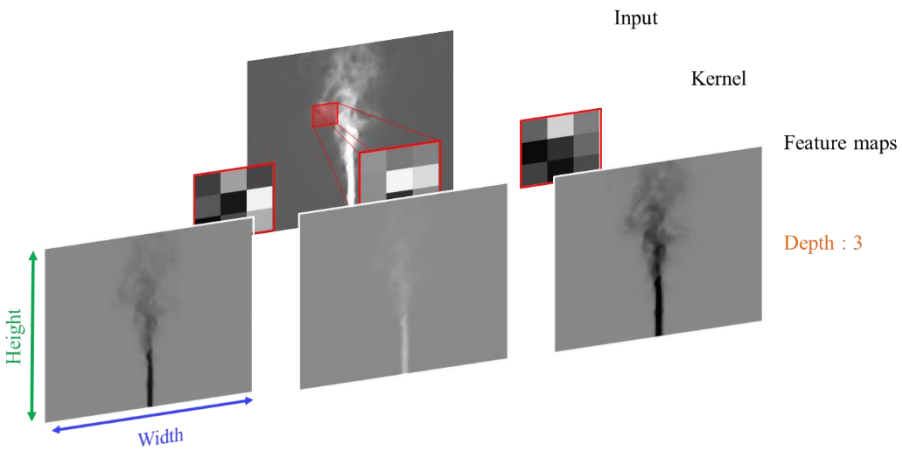

Figure 13. 2D convolution example using a kernel size of 3, stride of 1

This process is be expressed as using an input (I), kernel (K), and the output image (O):

$$O(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (36)$$

The output data has a smaller size than the input data in CNN because of combining multiple input data into a single output data. Thus, padding is used to maximize the utilization of edge data and prevent image compression. Mainly, same padding which make the out data in the same size as the input data is used.

In addition to padding, the output size is also affected by the stride and kernel sizes. When stride is set to 1, for instance, the kernel advances by one unit (Figure 14a). If the stride is 2, the kernel shifts two pixels, and the output size is decreased by one pixel relative to the previous example (Figure 14b).

In Figure 14c, the kernel size has been increased to 3, and as a result, the output data size is smaller than in Figure 14a. Accordingly, each convolutional layer can specify kernel size, stride, and padding, and must be adequately specified with output size in mind.

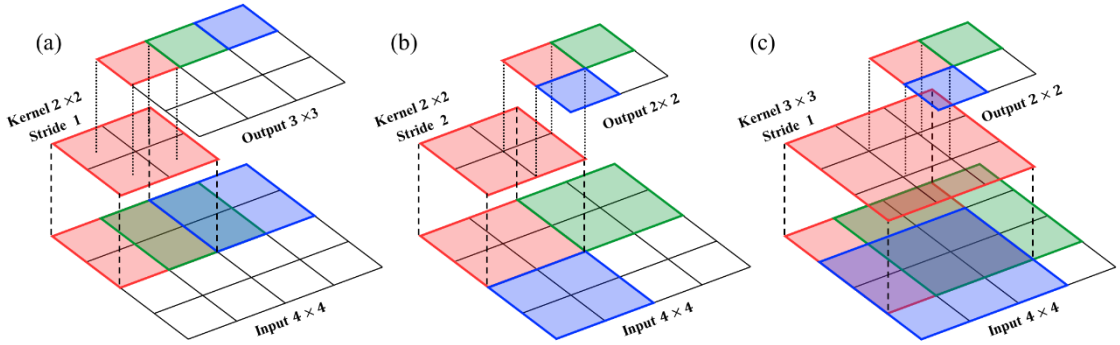

Figure 14. Convolution with kernel 2 and stride 1 (a); kernel 2 and stride 2 (b); kernel 3 and stride 1 (c)

A convolutional layer can be defined (Jin et al., 2018):

$$y_c = F(\kappa * x_c + b_c) \tag{37}$$

where y_c represents the output feature map, x_c the input feature map, * the convolutional operator, κ the learnable convolutional kernel, b_c the additive bias, F the nonlinear activation function.

After using convolutional layers, pooling is a popular layer in the CNN model to reduce the use of parameters and suppress over-fitting (Goodfellow et al., 2016). The pooling has an advantage known as 'Invariance to translation'. If only one data is extracted from a large number of input data, it means the extracted data still contains all of the input data's information. For example, if we translate whether an image has a human face or not, we only need to know whether the image has two eyes, nose, and a mouth; we do not need the exact location of the eyes, nose, and mouth. Therefore, the pooling uses smaller data sets which retain the characteristics of the entire data set. So, it is helpful to improve the

computational efficiency of the network. There are two representative pooling methods, average pooling and max pooling. They extract, respectively, the average and maximum value of a specified sub-region chosen by kernel. Figure 15 illustrates an example of an average and maximum pooling layer with stride of two pixels and kernel of two pixels. The output of the average pooling layer indicates the mean value of the selected data, whereas the output of the maximum pooling layer indicates the maximum value of the selected data.

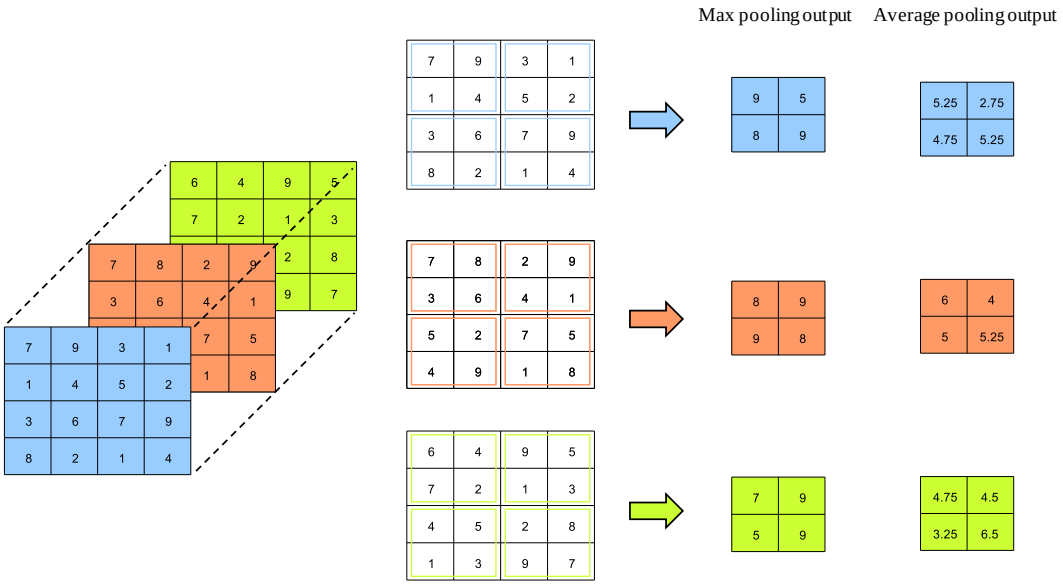

Figure 15. Example of average and maximum pooling layers with a stride of two pixels and a kernel of two pixels; the maximum and average values of the selected subregion are respectively calculated.

2.2.3 Generative Adversarial Network

Goodfellow introduced the Generative Adversarial Network (GAN) as a representative unsupervised model in 2014. (Goodfellow et al., 2014). A standard GAN model consists of a generator and discriminator pair network (Figure 16). Typically, these generator and discriminator networks are

comprised of convolutional and fully neural networks (Creswell et al., 2018). Generator creates an image from the random noise in order to create a target image, and discriminator receives both the real target image and the fake image created by the generator and determines which is real and which is fake. The error between the real and fake images motivates the generator and discriminator produce more realistic images and distinguish between real and fake images, respectively. Following training iterations, generative models are led to produce the fake image which is similar to real image, and discriminator models are eventually unable to distinguish between real and fake images.

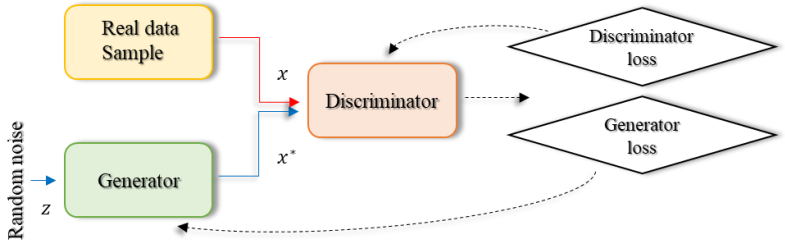

Figure 16. Typical model of generative adversarial network which has generator and discriminator; generator make the fake image from random noise and discriminator distinguish the real and fake images. The losses lead generator and discriminator make realistic image and find fake image well.

As the training iterations progress, the generator's distribution, p_g, which is specified by the input noise variables, $p_z(z)$, shifts in the direction of becoming more similar to the data distribution, p_{data}, in traditional GAN model. The mapping process to data space and parameter in GAN algorithm are denoted by $G(z)$ and θ_g, respectively. $D(x)$ is defined as the probability and changed depending on where x comes from the generator or real data. $D(x)$ is close to 1 when x come from the data and close to 0 otherwise. The Gans' loss function, $V(G,D)$, is written as equation 21 and it is known as two-player minmax gam (Goodfellow et al., 2014). The first term represents the expected value of the output when the real data x is input into the discriminator, and the second term is the expected value

when the fake data z is input into the discriminator after being input into the generator. When D discriminates has good performance, $D(x)$ becomes 1 and then the first term disappears. The samples created by $G(z)$ can be identified as fake values. Therefore $\mathrm{D}(G(z))$ can achieve 0 if it discriminates effectively. Also, because G is so proficient at generating data similar to real data, D does not recognize the image generated by G as fake and concludes that it is real in the second term. So, *G* has the minimum value negative infinity. In other words, this model trains *G* to minimize and *D* to maximize the loss function, respectively. The loss function is expressed as below and $\mathbf{E}$ stands for expected value.

$$minmaxV(G,D) = \boldsymbol{E}_{x\sim p_{data}(x)}[logD(x)] + \boldsymbol{E}_{z\sim p_z(z)}[log\,(1 - D(G(z)))] \quad (38)$$

The process of training is illustrated in Figure 17. In the beginning, *G* and *D* perform poorly, so generative distribution (red dotted line) is far from the data set distribution (black solid line) and discriminative distribution (green dashed line) is unable to distinguish the data (Figure 17a). As the GAN is being trained, *G* and *D* begins to be optimized. The learning rate of two networks are different (Goodfellow et al., 2014). Usually, *D* and *G* are optimized by k steps and 1 step, respectively. It means that *D* trains faster than *G*, so *D* can distinguish whether data come from data sets or samples. Therefore, at the middle of training, while *G* 's performance is poor, *D* distinguishes the data generated by *G* from real data samples too well (Figure 17b). Afterward, *G* is trained so that the data set distribution and generative distribution are similar to produce similar samples to the data (Figure 17c). After training Gan networks iterate, *D* is unable to distinguish the data because p_g is same as p_{data} (Figure 17d). So, *D(x)* yields only 0.5. This Gan algorithm has the benefit of being compatible with numerous models and requiring no inference during training. However, *D* and *G* should be well balanced and improved, it has the disadvantage that $p_g(x)$ does not explicitly exist (Goodfellow et al., 2014).

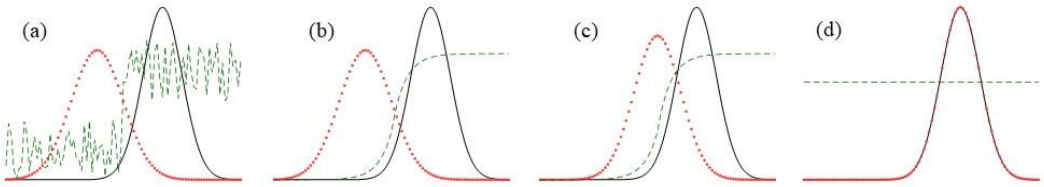

Figure 17. Several steps of GAN training by generative distribution (red dotted line), data set distribution (black solid line) and discriminative distribution (green dashed line); (a) generative distribution is different from data set distribution and discriminator is hard to differentiate between both distribution, (b) training of discriminator is faster than that of generator, so discriminate recognize generative and data set distribution well, (c) generative distribution is modified to match real data distribution, and (d) generative distribution is same as data set distribution and discriminator is incapable to differentiate both distribution

If the training and time are enough, G is optimized when the generator's distribution, p_g, has same the data distribution, p_{data}. It was proved by Goodfellow in 2014 and reproved here (Goodfellow et al., 2014). For a fixed G, a discriminator D maximize the quantity V:

$$V(D) = \boldsymbol{E}_{x\sim p_{data}(x)}[logD(x)] + \boldsymbol{E}_{z\sim p_g}[log\ (1 - D(G(x)))] \tag{39}$$

$$V(D) = \int_x\ p_{data}(x)log\ (D(x))dx + \int_z\ p_g(z)log\ (1 - D(g(z)))dz \tag{40}$$

$$V(D) = \int_x\ p_{data}(x)\ log\big(D(x)\big) + p_g(x)log\ (1 - D(g(x)))dx \tag{41}$$

If p_{data} and p_g change as a and b, respectively, the equation is as below:

$$V(D) = alog(D) + blog(1 - D) \tag{42}$$

This equation has a maximum value in $[0,1]$ when $D_O = \frac{a}{a+b} = \frac{p_{data}}{p_{data}+p_g}$. Therefore, a loss value is as follow:

$$V(G,D) = V(G,D_O) = \int_x\ p_{data}(x)\ log\big(D_O(x)\big) + p_g(x)log\ (1 - D_O(x))dx \tag{43}$$

$$= \int_x \; p_{data}(x)\, log\left(\frac{p_{data}}{p_{data}+p_g}\right) + p_g(x)\, log\left(1 - \frac{p_{data}}{p_{data}+p_g}\right) dx \tag{44}$$

$$= \int_x \; p_{data}(x)\, log\left(\frac{p_{data}}{p_{data}+p_g}\right) + p_g(x)\, log\left(\frac{p_g}{p_{data}+p_g}\right) dx \tag{45}$$

$$= \boldsymbol{E}_{x\sim p_{data}(x)}\left[log\frac{p_{data}}{p_{data}+p_g}\right] + \boldsymbol{E}_{x\sim p_g(x)}\left[log\frac{p_g}{p_{data}+p_g}\right] \tag{46}$$

The global optimum of the generator is achieved if $p_{data} = p_g$ and at that time a global minimum of loss function V_G is $-log4$. The equation 29 can be reformulated as:

$$\boldsymbol{E}_{x\sim p_{data}(x)}\left[log\frac{p_{data}}{p_{data}+p_g}\right] + \boldsymbol{E}_{x\sim p_g(x)}\left[log\frac{p_g}{p_{data}+p_g}\right] + log4 - log4 \tag{47}$$

$$= \boldsymbol{E}_{x\sim p_{data}(x)}\left[log\, 2\frac{p_{data}}{p_{data}+p_g}\right] + \boldsymbol{E}_{x\sim p_g(x)}\left[log\, 2\frac{p_g}{p_{data}+p_g}\right] - log4 \tag{48}$$

Kullback–Leibler divergence (KLD) is a function used to calculate the difference between any two probability distributions and it can be expressed as follows:

$$KLD(P\|Q) = \boldsymbol{E}_{\mathrm{x}\sim P}\left[log\frac{P}{Q}\right]. \tag{49}$$

So, the global minimum of loss function is arranged:

$$V_G = \boldsymbol{E}_{x\sim p_{data}(x)}\left[log\frac{p_{data}}{(p_{data}+p_g)/2}\right] + \boldsymbol{E}_{x\sim p_g(x)}\left[log\frac{p_g}{(p_{data}+p_g)/2}\right] - log4\sqrt{a^2+b^2} \tag{50}$$

$$= KLD\left(p_{data}\|\frac{p_{data}+p_g}{2}\right) + KLD\left(p_g\|\frac{p_{data}+p_g}{2}\right) - log4 \tag{51}$$

$$= 2JSD\left(p_{data}\|p_g\right) - log4 \tag{52}$$

Jensen– Shannon divergence (JSD) is a function between the model's distribution and the data generating process and it can be expressed as follows:

$$JSD(P\|Q) = \frac{1}{2}KLD\left(P\|\frac{P+Q}{2}\right) + \frac{1}{2}KLD\left(Q\|\frac{P+Q}{2}\right). \tag{53}$$

If the two distributions, p_g and p_{data}, are same, JSD yields 0. On the other hand, JSD becomes positive value. Therefore, the global minimum loss function, V_G, is $-log4$ if and only if $p_g = p_{data}$ (Goodfellow et al., 2014). At this moment, the discriminator just predicts 0.5 for all samples drawn from x. In other words, the generator, G, is optimal when the discriminator, D, cannot distinguish real data and fake samples.

2.3 Computational Fluid Dynamics

The motion of Newtonian fluids is governed by the nonlinear partial differential equations known as Navier-Stokes equations. These equations are used to describe a wide range of phenomena, such as weather patterns, ocean currents, and pipe flows (Galdi, 2011). Computational Fluid Dynamics (CFD) is a powerful tool for approximating solutions to these partial differential equations. In this technique, the PDEs are transformed into algebraic equations, which are then discretized by computers to find solutions. The continuity and momentum equations that describe fluid motion are given by Jiang and Lai (2016) as follows:

Continuity equation,

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{54}$$

Momentum equation,

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}\left(u_i u_j\right) = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 u_i}{\partial x_j \partial x_i} + g_i \tag{55}$$

where v, t, ρ, and g represent velocity, time, density, pressure, and gravity, respectively. Additionally, indices i = 1, 2, and 3 correspond to streamwise, spanwise, and vertical directions, respectively. The equation corresponds to unsteady, incompressible, and viscous fluid flows. These equations describe unsteady, incompressible, and viscous fluid flows.

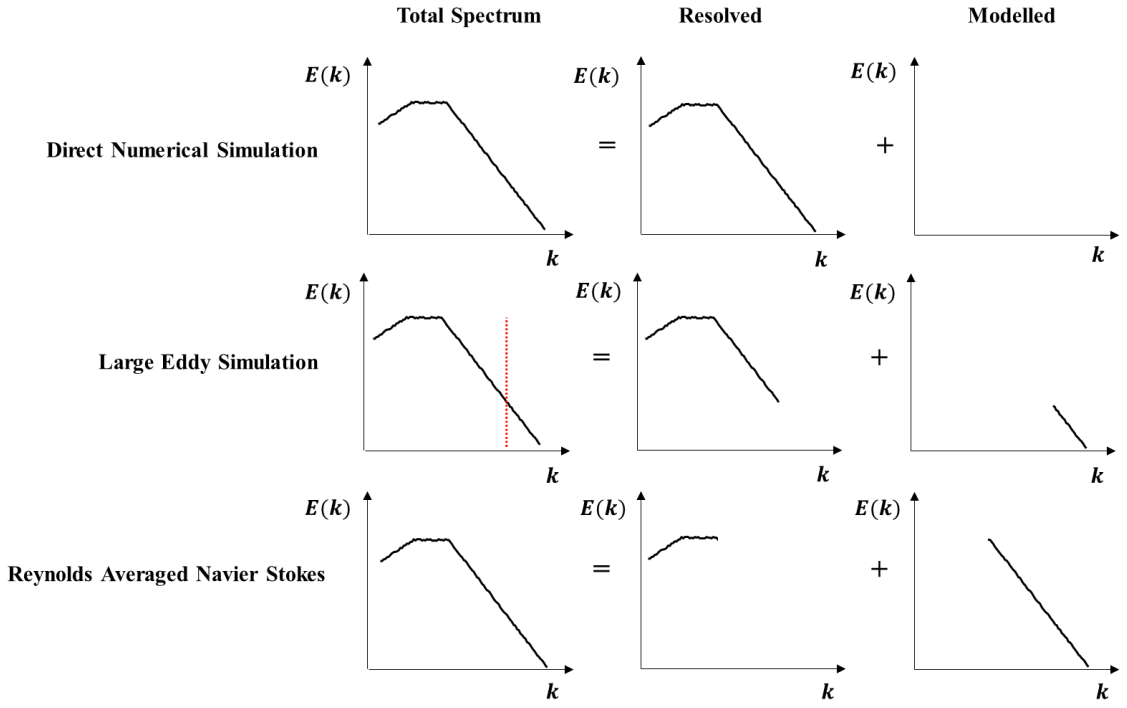

Figure 18. Decomposition of energy spectrum depending on computational numerical models; Direct numerical simulation (DNS), Large eddy simulation (LES) and Reynolds averaged Navier Stokes (RANS). DNS resolves all scales of eddies directly, while RANS models all scales of eddies. LES resolve the larger-scale eddies directly and requires artificial modeling for the remaining smaller-scale eddies.

Turbulence one of the most common types of irregulars, unstable, and random flows, and is characterized by its transport and mixing properties (Pope, 2000). To investigate these properties, there have been numerous theoretical, experimental, and computational studies on turbulence. In this study, turbulent flows were investigated using computational fluid dynamics (CFD), that employs three major turbulence models: DNS, LES, and RANS models. DNS solve the Navier-Stokes equations to obtain for one realization of the flow, resolving all length scales and timescales directly. As a result, DNS is computationally expensive. LES solves the equations for the filtered velocity field, which are equivalent to the large scales of the turbulence. The smaller-scale motions are not resolved directly and are instead

modeled. In RANS, the equations are solved for the mean velocity field, and a turbulence model based on either the turbulent viscosity hypothesis or the Reynolds-stress transport equation is used to model the Reynolds stresses (Pope, 2000). According to the turbulent-viscosity hypothesis in RANS, the Reynolds stresses are given

$$< u_i u_j >= \frac{2}{3} k \delta_{ij} - \nu_t (\frac{\partial < U_i >}{\partial x_j} + \frac{\partial < U_j >}{\partial x_i}). \quad (56)$$

Furthermore, the turbulent viscosity ν_t is calculated as the product of a velocity and a length scale.

$$\nu_t = u^* l^* \quad (57)$$

These are illustrated in Figure 18.

2.3.1 Turbulent Characteristic

The turbulent flows are characterized by the following properties (Ferziger, 2002):

• Turbulent flow is unsteady, meaning that the time series velocity plot exhibits randomness or unfamiliarity, and can be described as chaotic.

· Turbulent flows are three-dimensional, so the instantaneous field of turbulent flows has three spatial dimensions.

· As the intensity of turbulence increases, vortex stretching becomes one of the principal mechanisms. it means turbulent flows have a large amount of vorticity.

· Turbulent flows have coherent structures that are responsible for large-scale motion. However, the random component of turbulent flows results in divergent size, strength, and time scales, making their study difficult.

· Turbulent flows exhibit a broad range of length, time, and strength scales.

Turbulent eddies can have various sizes ranging from the width of the flow to smaller scales, and the smallest size is related to the Reynolds number (Pope, 2000). At high Reynolds numbers, turbulence exhibits two important characteristics: energy cascade and Kolmogorov scales. In 1922, Richardson first identified eddies of different sizes and the energy cascade process that occurs between them (Richardson, 1992). Large eddies contain smaller eddies, which have their velocity dissipated by viscosity until they fragment into even smaller eddies. This fragmentation process is known as the energy cascade. The size, velocity, timescale, and Reynolds number of the largest eddies are defined as l_L, $u(l_L)$, $\tau(l_L)$ and $\mathrm{Re}_L = u(l_L)l_L/\nu$. In large eddies, the Reynolds number is high, so the viscosity is negligible. This means that diffusion of momentum and energy is less effective, allowing for the transfer of energy to smaller scales through the energy cascade process (Richardson, 1922). Consequently, the smaller eddies exhibit a lower Reynolds number and a more stable motion with the molecular viscosity dominating. Ultimately, the energy transfer to smaller scales culminates in dissipation.

It is difficult to find a criterion to distinguish between large or small eddies. Three hypotheses suggested by Kolmogorov in 1941 can help to solve this problem (Kolmogorov,1941). The first hypothesis is that at sufficiently high Reynolds number, the small-scale turbulence is statistically isotropic (Pope, 2000). It is known as Kolmogorov's local isotropy hypothesis. Therefore, the criterion, l_{EI}, which separates anisotropic from isotropic motions, is useful. When the size is smaller than l_{EI}, the turbulent motion is isotropic. The second hypothesis is Kolmogorov's first similarity hypothesis. At a high Reynolds number, ν and ε determine the universal form of small-scale turbulent motion ($l < l_{EI}$) (Pope, 2000). The time scale of small size turbulent motion, $l/u(l)$, is smaller that of large size turbulent motion, $l_L/u(l_L)$. It means the small eddies quickly reach a dynamic equilibrium. In addition, the small eddies have unique length (η), velocity (u_η) and time scale (τ_η) determined by ν and ε (Kolmogorov,1941):

$$\eta \equiv (\nu^3/\varepsilon)^{1/4} \tag{58}$$

$$u_\eta \equiv (\varepsilon\nu)^{1/4} \tag{59}$$

$$\tau_\eta \equiv (\nu/\varepsilon)^{1/2} \tag{60}$$

The Reynolds number of Kolmogorov number is unity ($\eta u_\eta/\nu = 1$). If the turbulent motion is in Kolmogorov scale, it can be explained using the similarity hypothesis and the universal form.

The third hypothesis is Kolmogorov's second similarity hypothesis (Pope, 2000). The statistics of motion have a universal form determined by ε if the size is in $l_\eta \gg l \gg \eta$ at high Reynolds number. A length scale, l_{DI}, is the dividing standard between two parts. One part is affected by ν and ε, while the other section is influenced by only ε. The dissipation and inertial ranges are their respective part name. In inertial subrange ($l_{EI} > l > l_{DI}$), the motion is affected by ε, and the viscosity effects is neglected. On the other hand, the viscosity is a significant effect in dissipation rate. These various length scales and ranges are summarized in Figure 19. In inertial subrange, the velocity scales and time scales are denoted (Pope, 2000):

$$u(l) = (\varepsilon l)^{1/3} = u_\eta (l/\eta)^{1/3} \sim u_L (l/l_L)^{1/3} \tag{61}$$

$$\tau(l) = (l^2/\varepsilon)^{1/3} = \tau_\eta (l/\eta)^{2/3} \sim \tau_\eta (l/l_L)^{2/3} \tag{62}$$

As shown in equations, the velocity and time scale decrease as the length, l, decrease. The quantity of energy transported from the large to small eddies is denoted by $E_t(l)$ and has an order of magnitude of $u^2(l)/\tau(l)$.

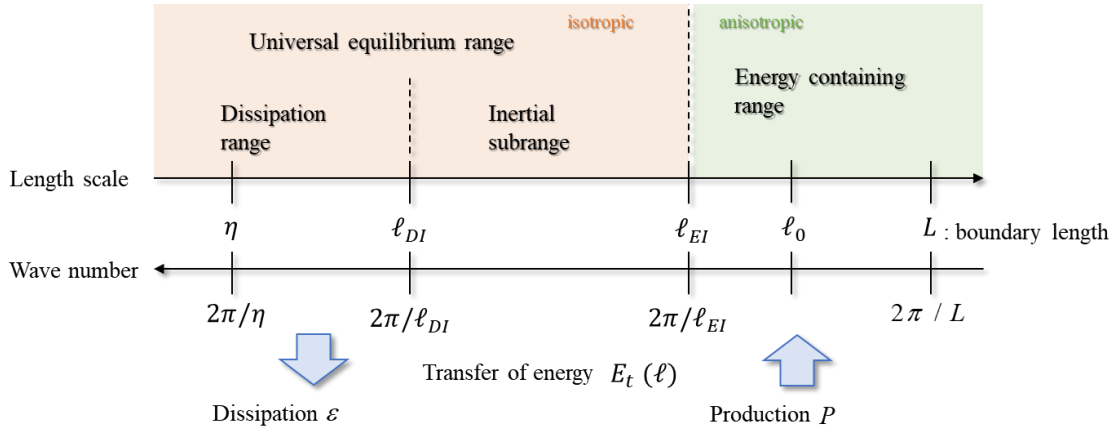

Figure 19. Schematic diagram of various length scale, wave number and ranges. The energy containing range product the energy and transferred energy dissipated in dissipation range. The transfer of energy is called as energy cascade and order is $u^2(l)/\tau(l)$

A turbulent flow can achieve a statistically stationary state after an initial transient period. The autocovariance ($R(s)$) for a statistically stationary state be defined as

$$R(s) =< u(t)u(t+s) > \tag{63}$$

and the autocorrelation function can be normalized as

$$\varrho(s) =< u(t)u(t+s) >/< u(t)^2 > \tag{64}$$

where $u(t)$ is fluctuation of velocity at time t. The autocorrelation function is symmetric about $s = 0$, meaning that $\varrho(s) = \varrho(-s)$. Additionally, the autocorrelation function satisfies the following conditions:

$$\varrho(0) = 1,\ \varrho(s) \leq 1 \tag{65}$$

Here, $\varrho(0)$ represents the correlation of fluctuations at the same time, which is always 1. The autocovariance is defined as $R(s) \equiv< u(t)^2 > \varrho(s)$, and the frequency spectrum $E(\omega)$ is a pair with $R(s)$ as its Fourier transform.

$$E(\omega) \equiv \frac{1}{\pi}\int_{-\infty}^{\infty} R(s)e^{-i\omega s}ds \quad (66)$$

$$= \frac{2}{\pi}\int_{-\infty}^{\infty} R(s)cos\,(\omega s)ds \quad (67)$$

$$R(s) \equiv \frac{1}{2}\int_{-\infty}^{\infty} E(\omega)e^{i\omega s}d\omega \quad (68)$$

$$= \int_{-\infty}^{\infty} E(\omega)\,cos(\omega s)\,d\omega. \quad (69)$$

2.3.2 Direct Numerical Simulation

Direct Numerical Simulation is a model used to solve Navier-Stokes equations without averaging or approximation, making it the most accurate approach (Ferziger et al., 2002). For useful results, the computational domain should be larger than the physical domain or the largest turbulent eddy. Additionally, it can capture the kinetic energy dissipation if the results are valid. The dissipation occurs on the smallest scales, and viscosity is relevant in this process. Therefore, the grid size must be smaller than the Kolmogorov scale, η (Lesieur, 1987). To produce accurate results using DNS, the length and time scales are important. The number of grids in each direction must be at least L/η, where η is the Kolmogorov length scale defined as $\eta \equiv (\nu^3/\varepsilon)^{1/4}$, for homogeneous isotropic flow that use the uniform grid. The boundary length is almost similar to the largest eddies, so the number of grids can be expressed as l_L/η. The dissipation order is $u^3(l_L)/l_L$, and the number of grids in one direction can be rewritten as $(u(l_L)l_L/\nu)^{3/4}$. Therefore, the total number of grids in three directions is proportional to $\mathrm{Re}^{9/4}$. As the spacing between grids is fine in DNS, the time step must also be fine. Previous studies have shown that to produce accurate results, the number of time steps should be proportional to $\mathrm{Re}^{3/4}$ (Ferziger et al., 2002). However, the large number of grids required in DNS means that a lot of time steps are also necessary. Taking into account the overall number of grids and time steps, it can be seen that they are proportional to Re^6. Because the number of grids and time steps can become extremely

large, DNS has many limitations in solving fully turbulent models at high Reynolds numbers. Simple flows at low Reynolds numbers can be simulated using DNS on workstations. However, the required number of grid points and computational time depend on the specific flow and Reynolds number being simulated.

DNS requires accurate numerical methods to produce reliable results for flows that have a wide range of time and length scales. In most cases, explicit methods that are stable for the required time step based on the accuracy requirement are available and are commonly used (Ferziger et al., 2002). However, there is a notable exception when dealing with solid surfaces, as the important structures in these regions are of very small size, and very fine grids must be used, especially in the direction normal to the wall. In these cases, implicit methods may be necessary to maintain numerical stability, even though they come with an extra computational cost. The most commonly used time methods are those of second to fourth order accuracy. Among them, Runge-Kutta methods have been used most frequently, although other methods are also used (Ferziger et al., 2002). While Runge-Kutta methods require more computation per time step than other methods of the same order, they are preferred because they produce much smaller errors for a given time step.

DNS resolves a wide range of length scales, requiring an accurate discretization method. The velocity fields can be expressed using Fourier transforms, and the wavenumber resolved is $\pi/\Delta x$:

$$u(x) = \sum \tilde{u}(k)\, e^{ikx}. \tag{70}$$

The exact derivative of e^{ikx}, ike^{ikx} can be replaced by $ik_{eff}e^{ikx}$ where k_{eff} is the effective wavenumber. The difficulty encountered in simulating turbulent flows is that the turbulence spectra, which describe the distribution of turbulence energy over wavenumber or inverse length scale, are typically large over a significant range of the wavenumber spectrum 0 to $\pi/\Delta x$. As a result, a better measure of error is:

$$\epsilon^2 = \frac{\int (k - k_{eff})^2 E(k) dk}{\int k^2 E(k) dk}. \quad (71)$$

Here, $E(k)$ is the energy spectrum of the turbulence in one dimension, and it is calculated as $\hat{u}(k)\hat{u}^*(k)/2$.

2.3.3 Large Eddy Simulation

Large eddy simulation (LES) resolves the large-scale motions directly but parameterizes small scales by modeling them (Piomelli, 1999). The smaller eddies are assumed to be homogeneous and universal, and LES emerged to model their motion. LES takes less time than DNS, which solves all scales of eddies directly for the same flow, but it is still expensive and time-dependent (Sagaut, 2006). As the Reynolds number increases, the LES method is preferred over DNS. When LES is used to simulate the flow, the separation of large and small eddies is important. Usually, a filtering operation of the velocity is defined as follows (Ferziger, 2002):

$$\widetilde{u_i}(x) = \int G(x, x') u_i(x') dx' \quad (72)$$

where G is a filter function which determine the range of small scale. The preferred filter functions are the sharp Fourier cutoff filter, Gaussian filter, and top-hat filter, respectively. They are defined as (Piomelli, 1999):

$$G(k) = \int G(x') e^{-ikx'} dx' = \begin{cases} 1 & if\ k \le \pi/\bar{\Delta} \\ 0 & ohterwise \end{cases}, \quad (73)$$

$$G(x) = \sqrt{\frac{6}{\pi \bar{\Delta}^2}} exp\left(-\frac{6x^2}{\bar{\Delta}^2}\right), \quad (74)$$

$$G(k) = \begin{cases} 1/\tilde{\Delta} & if\ |x| \le \tilde{\Delta} \\ 0 & ohterwise \end{cases}. \quad (75)$$

In these equations, $\tilde{\Delta}$ represents the filter width. In OpenFOAM, the top-hat filter is applied as a

standard filter, and the filter width is usually set to the characteristic length of grid size (Sagaut, 2005). Figure 20 shows that eddies larger than the grid size are considered as resolved directly and are classified as large-size eddies. Conversely, small eddies, which are smaller than the grid size, are modeled.

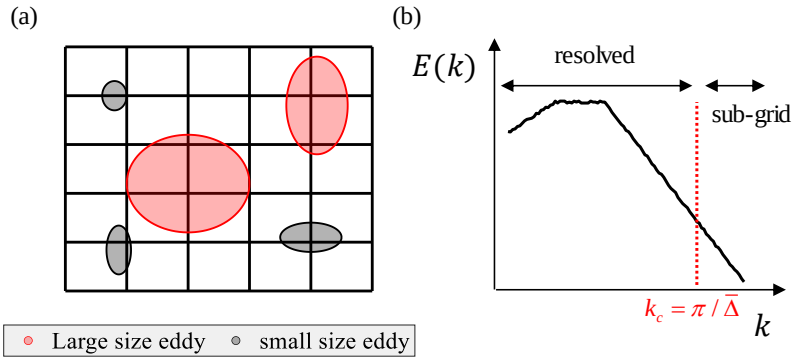

Figure 20. (a) The schematic image in the domain illustrates large and small size eddies. Larger size eddies are larger than the grid size, while small size eddies are smaller than the grid. (b) The cutoff length and wavenumber are represented by $\bar{\Delta}$ and $\pi/\bar{\Delta}$, respectively. Thus, larger eddies whose wave number is larger than the cutoff wavenumber are resolved directly, while small eddies are modeled.

The continuity and momentum equations applied a filter:

$$\frac{\partial \widetilde{u_i}}{\partial x_i} = 0 \tag{76}$$

$$\frac{\partial \widetilde{u_i}}{\partial t} + \frac{\partial (\widetilde{u_i}\widetilde{u_j})}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \tilde{p}}{\partial x_i} + \nu \frac{\partial}{\partial x_j}\left(\frac{\partial \widetilde{u_i}}{\partial x_j} + \frac{\partial \widetilde{u_j}}{\partial x_i}\right) + g_i. \tag{77}$$

The tilde symbol denotes the filtered operation applied to these equations. The velocity can also be decomposed as $u = \tilde{u} + u'$. In the momentum equation, the second term on the left side is non-linear and can be expressed as (Jawahar et al., 2018):

$$\widetilde{u_\iota u_J} = (\widetilde{u_\iota} + \widetilde{u_\iota})(\widetilde{u_J} + u_J) = \widetilde{\widetilde{u_\iota}\widetilde{u_J}} + \widetilde{\widetilde{u_\iota}u'_J} + \widetilde{u'_\iota\widetilde{u_J}} + \widetilde{u'_\iota u'_J}\, u \tag{78}$$

So, the filtered momentum equation is rewritten:

$$\frac{\partial\widetilde{u_\iota}}{\partial t} + \frac{\partial(\widetilde{u_\iota}\widetilde{u_J})}{\partial x_j} = -\frac{\partial\tilde{p}}{\partial x_i} + \nu\frac{\partial}{\partial x_j}\left(\frac{\partial\widetilde{u_\iota}}{\partial x_j} + \frac{\partial\widetilde{u_J}}{\partial x_i}\right) + \frac{\partial\tau_{ij}^{SGS}}{\partial x_j}. \tag{79}$$

The subgrid-scale stresses are denoted as τ_{ij}^{SGS}, and they can be decomposed into Leonard stress, $L_{ij} = \widetilde{\widetilde{u_\iota}\widetilde{u_J}} - \widetilde{u_\iota}\widetilde{u_J}$, cross terms, $C_{ij} = \widetilde{\widetilde{u_\iota}u'_J} + \widetilde{u'_\iota\widetilde{u_J}}$, and Reynolds subgrid-scale tensor, $R_{ij} = \widetilde{u'_\iota u'_J}$ (Piomelli, 1999). The Leonard stress represents the interaction between filtered values, and the cross term is associated with the interaction between resolved and modeled values. Lastly, the Reynolds subgrid-scale is related to the interaction between subgrid-scales. The resolved scale is almost unable to dissipate the energy, so the subgrid-scale removes the energy (Sagaut, 2006). The width of the scales can be defined using various methods, with the cubic method being one of the most commonly used. The grid spacings in the three coordinate directions are denoted by Δ_1, Δ_2, and Δ_3. A characteristic filter width Δ is taken to be $\Delta\equiv(\Delta_1\Delta_2\Delta_3)^{1/3}$.

There are various methods to resolve subgrid-scale stresses, including the Smagorinsky and dynamic models. Firstly, the Smagorinsky model assumes a reasonable model, given by Equation (52), where the subgrid-scale stresses τ_{ij}^{SGS} are related to the resolved scale strain-rate tensor of large scale ($\widetilde{S_{\iota J}}$) and eddy viscosity

$$\tau_{ij}^{SGS} - \frac{\delta_{ij}}{3}\tau_{kk}^{SGS} = \nu_{\mathrm{SGS}}\widetilde{S_{\iota J}}. \tag{80}$$

The form of the subgrid-scale eddy viscosity can be derived by dimensional arguments, yielding (Ferziger, 2002):

$$\nu_t = C_s^2\Delta^2|\widetilde{S}| \tag{81}$$

where C_S is a model parameter to be determined, $|\widetilde{S}| = \left(\tilde{S}_{ij}\tilde{S}_{ij}\right)^{1/2}$. For isotropic turbulence, the value

of C_s is typically taken as 0.2, but it needs to be reduced to approximately 0.065 in the bulk of the flow (Ferziger, 2002).

The dynamic models compute the model parameters at every spatial grid point and every time step from results of the LES. It was first introduced by Germano et al. (1991) in detail. These dynamic models solve several difficulties that arise with the use of traditional eddy viscosity models (Ferziger, 2002). For instance, in shear flows, the Smagorinsky model parameter needs to be much smaller than in isotropic turbulence. The dynamic model produces this change automatically, eliminating the need for manual adjustment. Additionally, the model parameter needs to be reduced even further near walls, and the dynamic model automatically decreases the parameter in the correct manner near the wall. Another issue with traditional models is the unclear definition of the length scale for anisotropic grids or filters, but this problem is resolved with the dynamic model since the model compensates for any error in the length scale by changing the value of the parameter.

In LES, the grid size is an important factor, similar to DNS. The grid size takes into account the characteristic length, denoted by l_0. The characteristic length is the largest size resolved in LES. To ensure that 80% of the total energy is resolved, the grid spacing in all three dimensions should be smaller than l_{EI} (Pope, 2000), where l_{EI} is calculated as $l_0/6$. Specifically, l_{EI} is obtained by integrating the autocorrelation function $f(r,t)$ over the domain, given by:

$$l_{EI} = \frac{l_0}{6} = \int f(r,t)dr = \int < u_i'(x+r,t)u_i'(x,t) > / u_{RMS}' \, dr. \tag{82}$$

where u_{rms}' is root mean square velocity.

2.4 Fourier Transform

In numerous fields, the Fourier transform (FT) has been extensively used to solve numerous

engineering problems. FT is a technique for decomposing the original data into sinusoids of various frequencies (Hoffman, 1997). It is advantageous that it can divide any data into the sum of temporal or spatial signals. The two-dimensional FFT was developed so that these FT methods could be applied to images. An image can be described mathematically as some function $f(x,y)$, where x and y are spatial coordinates of image and $f(x,y)$ represents the data of the image at point (x,y). So, the two-dimensional Fourier transform, and the inverse Fourier transform are defined below (Rzeszotarski's work,1983):

$$F(u,v) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-i2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{83}$$

$$f(x,y) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F(u,v)\, e^{i2\pi(\frac{ux}{M}+\frac{vy}{N})}. \tag{84}$$

If the original image, $f(x,y)$, has dimensions M by N, then the x and y have ranges of 0 to $M-1$ and 0 to $N-1$, respectively. i equals the square root of -1 and u and v are spectral coordinates. With FT's aforementioned characteristics, the original image is transformed into a linear combination of multiple frequencies (Burger & Burge, 2016).

$$e^{i2\pi(\frac{ux}{M}+\frac{vy}{N})} = \cos\left[2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right] + isin[2\pi(\frac{ux}{M}+\frac{vy}{N})] \tag{85}$$

Equation 2 shows two-dimensional cosine and sine functions with horizontal and vertical wave numbers u and v, respectively, and the corresponding angular frequencies w_m, w_n. At this time, the shape of the Fourier Transform domain changes depending on the values of M and N. Figure 21 depicts how the Fourier domain changes by setting various values of wavenumber m and n in horizontal and vertical directions. First, when both M and N are zero, there is no specific frequency, so the domain is flat. Similarly, as M and N change, the domain takes on a specific shape. If M is zero, the frequency only exists along the y direction. In contrast, when N is zero, there is only x axis frequency. As the values of M and N increase, the wave's form becomes increasingly complex.

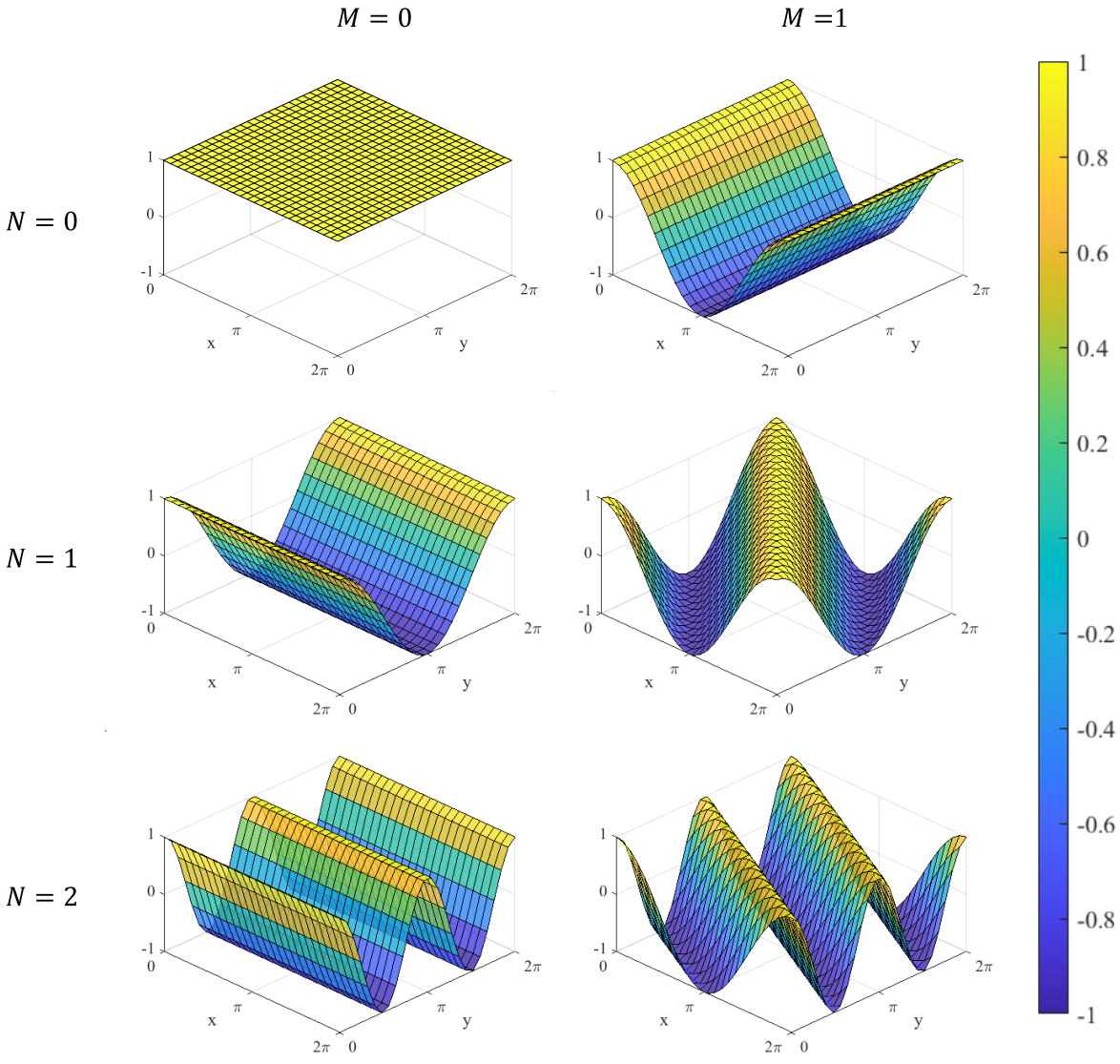

Figure 21. Two-dimensional Fourier domain of cosine functions by wave number M and N in horizontal and vertical directions

The Fourier transform, $F(u,v)$, is complex numbers in general, $F(u,v) = F_R(u,v) + F_I(u,v)$. $|F(u,v)|$ is the magnitude spectrum and it is calculated $\sqrt{F_R(u,v)^2 + F_I(u,v)^2}$. Also, the phase angle, $\emptyset(u,v)$, is defined as $tan^{-1}\left[\frac{F_I(u,v)}{F_R(u,v)}\right]$. The two -dimensional Fourier Transform has two basic properties, periodicity and symmetry (Burger & Burge, 2016). First, periodicity is that the spectrum repeats in one

direction by M and N, respectively.

$$F(u,v) = F(u + pM, v + qN) \text{ where } \mathrm{p,q} \in \{-\infty, \dots, -1,0,1, \dots, \infty\}. \quad (86)$$

The second characteristics, symmetry, is symmetric at the origin and can be represented as

$$|F(u,v)| = |-F(u,v)|. \quad (87)$$

These characteristics are because the original data is a real number, but it is changed to a complex number when FT is applied. Therefore, it is common to move point, $(u,v) = (0,0)$, to the midpoint of the domain when drawing the FT domain. Figure 22 shows three examples of two-dimensional Fourier spectra of a rectangle with different angles. The angles of the rectangle are 0°, 30° and 45° respectively. First, when the angle of the rectangle is 0, the Fourier domain has two horizontal and vertical lines. When this rectangle rotates to the right, the two straight lines inside the Fourier domain also rotate to the right. As the angle of the rectangle increases, increases, the inclination of the lines in the spectrum domain increases.

Figure 23 illustrates the application of a two-dimensional Fourier transform to a more complex image. It is a photograph of the Han River in Seoul taken from Gwangjang-dong. The log-scale magnitudes of Fourier transform in original and shifted coordinates are illustrated in Figure 23 b and c, respectively. The log-scale Fourier transform is calculated as log(1+|F(u,v)|) to avoid that the number in logarithm is zero. As shown in Figure 23b, the frequency in one direction increases and decreases as far from the lower left corner. So, as mentioned earlier, it is recommended to shift the spectrum so that the center frequency is (0,0) like Figure 23c. Therefore, the Fourier transform mentioned in the future refers to a domain which (0,0) is shifted to the center, as shown in Figure 23c.

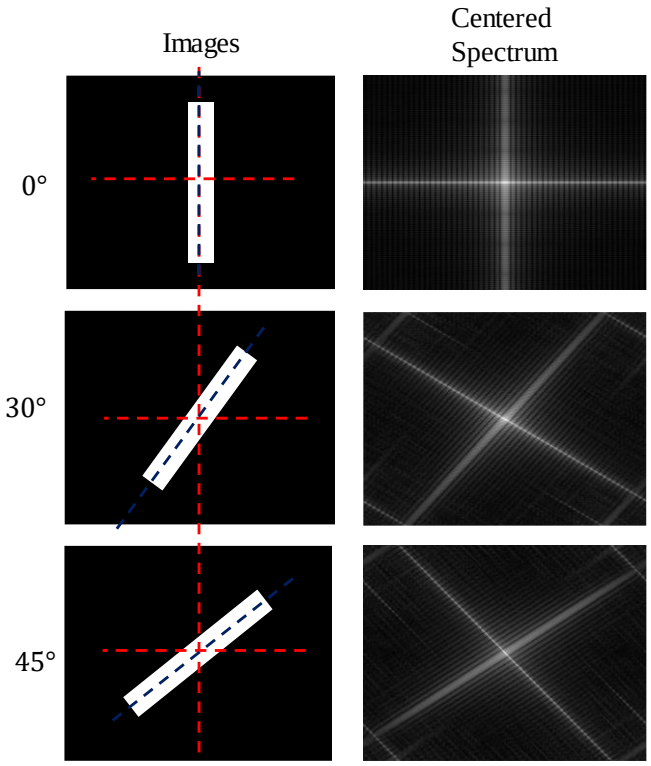

Figure 22. Three example of rectangle images with the various angles and its centered Fourier spectrums

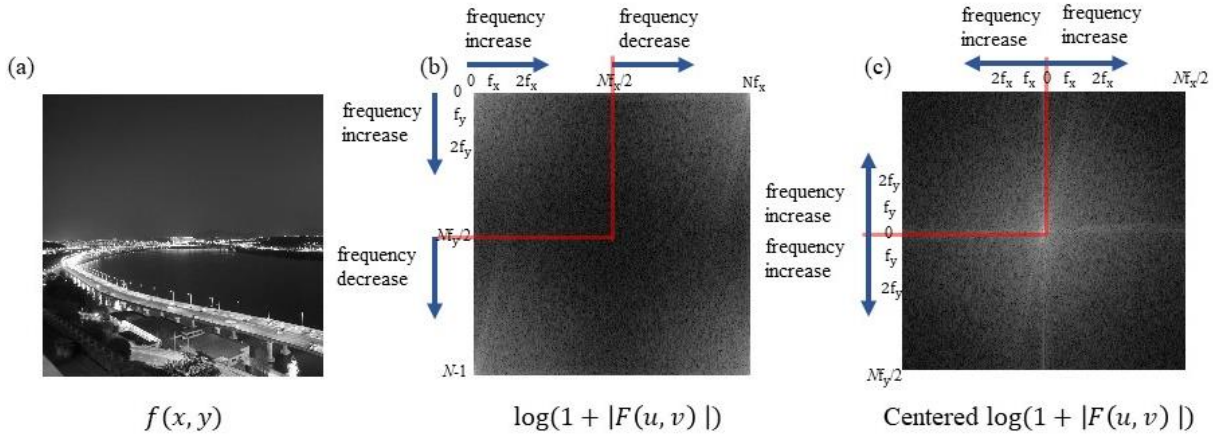

Figure 23. (a) Original image, (b) Fourier Transform and (c) shifted Fourier Transform

Filtering is used to eliminate the unwanted frequencies (Dogra et al., 2014). Typically, image

filtering is usually applied to remove noise and improves the digital image for varied application (Dogra et al., 2014). A filter is a matrix composed of values between 0 and 1. If the component is 1, the frequency remains constant, and if it is 0, the frequency is eliminated. The box filtering is an ideal and straightforward filtering method because it eliminates the range above or below the specified frequency. The definitions of low pass filter (H_L) and high pass filter (H_H) are

$$H_L(u,v) = \begin{cases} 1 & \sqrt{(u-M/2)^2+(v-N/2)^2} \le h_0 \\ 0 & \sqrt{(u-M/2)^2+(v-N/2)^2} > h_0 \end{cases} \tag{88}$$

$$H_H(u,v) = \begin{cases} 1 & \sqrt{(u-M/2)^2+(v-N/2)^2} \ge h_0 \\ 0 & \sqrt{(u-M/2)^2+(v-N/2)^2} < h_0 \end{cases} \tag{89}$$

Figure 24 illustrates the example of applying box filtering to the photograph of the Hangang River. Figure 24a and d show the original photograph and log-scale Fourier transform, respectively. When box filtering was performed, h_0 was designated as 30. The low pass filter leaves low frequencies to the spectrum, and the low pass filtered inverse Fourier transform has a loss of sharpness (Figure 24b and e). In contrast, a high pass filter eliminates low frequency, leaving the image with only edge information (Figure 24c). Typically, low pass filtering is used to smooth the image, while high pass filtering is used to sharpen the image (Dubey, 2014). This is because the edges or sharp parts of the image are converted to the sum of the high frequencies after Fourier transform (Dubey, 2014).

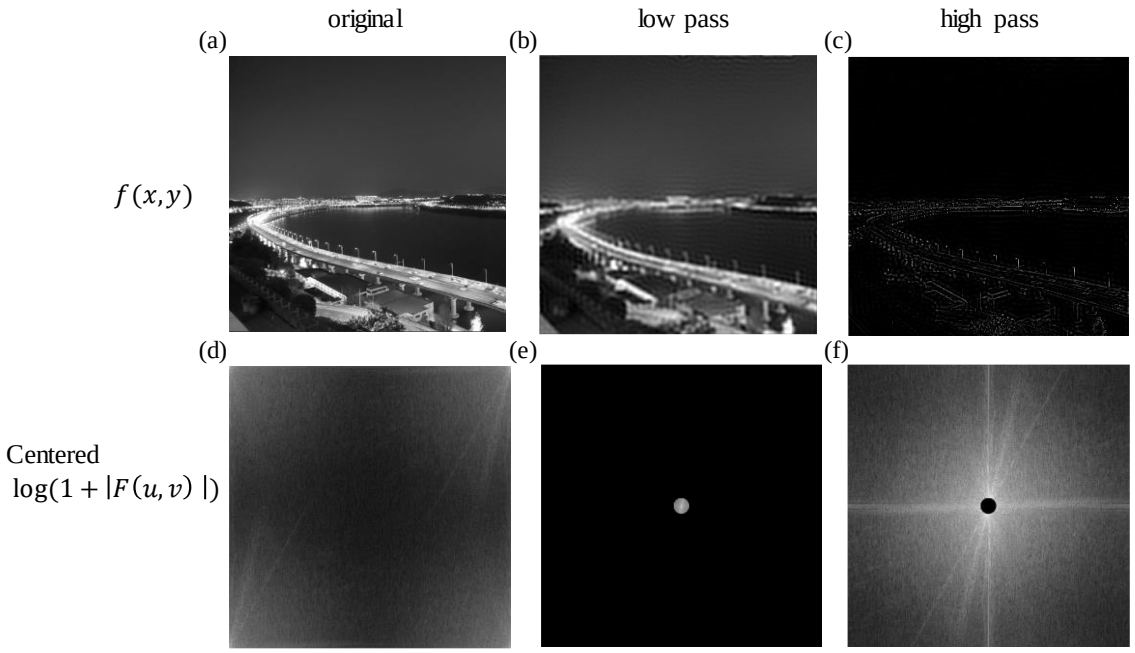

Figure 24. (a) Original photograph, (b) low pass filtered inverse Fourier transform, (c) high pass filtered inverse Fourier transform, (d) centered log-scale Fourier spectrum of original photograph, (e) log-scale Fourier spectrum after low pass filtering, and (f) log-scale Fourier spectrum after high pass filtering

CHAPTER 3. LITERATURE REVIEW

3.1 Jet Flow Literature Review

The jet flow has been extensively studied as a valuable method for analyzing turbulent flows. Early research conducted by Liepmann and Laufer (1947) focused on the average and turbulent flow characteristics of jet flows, as well as the mixing processes involved. Their work revealed the complex interactions, including energy exchange, that occur in the near-field region of jet flows. Building upon this knowledge, Matsuda and Sakakibara (2005) performed experimental visualization of three-dimensional turbulent flows in jet systems. They conducted experiments across various Reynolds numbers and successfully identified the vortex structures present in three-dimensional jet flows. Additionally, Shinneeb et al. (2008) investigated the large-scale structures within the near-field region of jet flow exits. They employed the Proper Orthogonal Decomposition (POD) method to analyze the rotational motion, size, and strength of these large vortices. Typically, the near field of a jet exit is defined as the region within approximately six times the jet diameter (Abdel, 2010). These studies collectively contribute to our understanding of the near-field behavior and characteristics of jet flows.

In the context of jet flows, the region located at a significant distance from the jet nozzle is commonly referred to as the far-field or fully-developed region (Ball et al., 2012). Numerous previous studies have proposed the applicability of similarity theory in the far-field region (Johansson et al., 2003; Revuelta et al., 2002; Gourlay et al., 2001). Johansson et al. (2003) divided the flow in front of the jet nozzle into two sections: the near wake region and the far wake region. They discovered that the flow in the near wake region does not exhibit equilibrium similarity, indicating a nonequilibrium state. However, they observed that the flow in the far wake region adheres to equilibrium similarity. Similarly, Gourlay et al. (2001) conducted numerical simulations using a DNS (Direct Numerical Simulation) turbulent model to investigate the velocity, turbulent amplitude, and wake velocity in jet flows. Their findings confirmed the presence of self-similarity in the late wakes of both unstratified and density

stratified fluids. Notably, they successfully captured the characteristics of late wakes, which develop in the far field, in both stratified and non-stratified fluids. These studies support the notion of equilibrium similarity in the far-field region of jet flows.

The presence of universal self-similarity in the far field of jet flows has been a topic of debate in previous studies (Uddin and Pollard, 2007; Xu & Antonia, 2002). This implies that the initial conditions can have an impact on the self-similar region (Abdel, 2010). Uddin and Pollard (2007) conducted LES (Large Eddy Simulation) simulations with various initial conditions. They manipulated the jet radius-to-width ratio and analyzed the mean velocity profiles and turbulence intensities in different cases. They observed that the mean velocity statistics followed the universal equilibrium theory, suggesting self-similarity. However, the turbulent statistics were found to be influenced by the initial conditions, indicating a departure from universal self-similarity. Similarly, Xu & Antonia (2002) performed experimental investigations by comparing two jet flows: one from a smooth contraction nozzle and the other from a pipe nozzle (fully developed turbulent profile). They analyzed the mean velocity, turbulent intensities, and Reynolds shear stress. Their findings revealed that the flow from the smooth contraction nozzle achieved self-similarity more rapidly compared to the long pipe flow. These studies highlight the influence of initial conditions on the self-similarity of jet flows, indicating that the attainment of universal self-similarity can be affected by specific factors such as nozzle geometry and initial flow configuration.

Furthermore, several previous studies have suggested that the flow in both the near and far field of jet flows is influenced by the Reynolds number (Deo et al., 2008; Kwon & Seo, 2005; O'Neill et al., 2004). Deo et al. (2008) conducted experiments on planar jets spanning a wide range of Reynolds numbers. Unlike previous studies that focused on Reynolds numbers above 10,000, they investigated Reynolds numbers ranging from 1,500 to 16,500, encompassing a much broader range. Their findings revealed that as the Reynolds number increases, the potential core of the jet becomes shorter and the spreading rate in the near field increases. However, in the far field, the spreading rate and velocity decay

decrease with increasing Reynolds number. Studies have also explored the effects of Reynolds number on not only circular jets but also planar jets. Kwon & Seo (2005) divided the jet flow into two sections: the zone of flow establishment and the zone of established flow, representing the near field and far field, respectively. They conducted experiments on round jets covering a wide range of Reynolds numbers. They observed that the dimensionless length of the zone of flow establishment decreases as the Reynolds number increases. The axial velocity profiles were found to closely follow a Gaussian distribution with increasing Reynolds number. O'Neill et al. (2004) performed PIV (Particle Image Velocimetry) experiments on round jet flows. They investigated the instability of the jet flow structure and identified certain Reynolds numbers, such as 680 and 1,030, where the flow structure exhibited clear instability. Furthermore, they noted the presence of distinct vortical structures near the orifice at both low and high Reynolds numbers. In summary, jet flows have served as representative cases in turbulent flow studies, and the turbulence of jets continues to be a subject of ongoing research. The study of jet flow is summarized in Table 4.

Table 4. Previous studies for jet flows

Papers	Methods	Buoyancy	Purpose
Gourlay et al. (2001)	CFD(DNS)	Simple jet and buoyant jet	Confirmed the vortices in unstratified and density stratified fluids proved that the mean late wakes which occur in far field has self-similarity
Xu & Antonia (2002)	Experiment(X-wire)	Simple jet	Compared the results of two jet flows, a smooth contraction nozzle and a pipe nozzle (fully developed turbulent profile). Found out that a smooth contraction nozzle had the self-similarity state more rapidly than a long pipe.
Johansson et al. (2003)	Experiment and CFD(DNS)	Simple jet	Divided the flow in front of jet nozzle into 2 sections named near wake region and far wake region and indicated that the flow in far wake region obeys the equilibrium similarity
O'Neill et al. (2004)	Experiment (PIV)	Simple jet	Conducted the PIV experiments of the round jet flow. Investigated the instability of jet flow

			structure and found that the jet flow structure was unstable at certain Reynolds numbers such as 680 and 1,030. Indicated that the flow near the orifice has clear vortical structures at both low and high Reynolds numbers
Matsuda & Sakakibara (2005)	Experiment (PIV)	Simple jet	Conducted the experiments in various Reynolds number and found out the vortex structures of three- dimensional jet flow. Visualized the three-dimensional turbulent flow
Kwon & Seo (2005)	Experiment (PIV)	Simple jet	Conducted the round jet experiments for various Reynolds number which has wide range. Found that the dimensionless length of the zone of flow establishment decreases as the Reynolds number increases. Checked the axial velocity profiles followed Gaussian distribution well as the Reynolds number increases.
Uddin and Pollard (2007)	CFD(LES)	Simple jet	Changed the rate of jet radius over width and obtained the mean and fluctuated velocity. Compared the mean velocity profile and turbulence intensity. Found out that the mean statistics followed the universal equilibrium theory, but turbulent statistics were affected by initial conditions.
Shinneeb et al. (2008)	Experiment (PIV)	Simple jet	Investigated large-scale vortical structures in the near field of jet flow exit and applied POD method to study position, size, strength and rotational sense of these vortices
Deo et al. (2008)	Experiment (hot-wire anemometry)	Simple jet	Conducted the plan jet experiments for various Reynolds number which has wide range. Indicated that as the Reynolds number increases, the potential core is getting shorter and the near-field spreading rate increases.

3.2 Machine Learning Literature Review

ML has gained significant popularity and is being applied to various tasks, including data reconstruction, turbulence modeling, and solving non-linear partial differential equations (PDEs). In the hydraulic field, there are three main topics where machine learning is utilized (image processing, making turbulence models and solving non-linear PDEs). In the area of image processing, ML

techniques have been employed to enhance low-resolution or noisy images (Dong et al., 2015; Maulik & San, 2017; Jin et al., 2018; Deng et al., 2019; Kim et al., 2021). Dong et al. (2015) proposed a deep CNN model for single-image super-resolution, achieving superior results with minimal preprocessing and postprocessing. The model demonstrated robustness and simplicity, as it was successfully tested on various cases. Similarly, Jin et al. (2018) utilized CNNs for image processing, specifically predicting time-series velocity fields from pressure fields at different Reynolds numbers. They proposed a three-path CNN architecture by extending the existing one-path CNN approach. Maulik & San (2017) developed a single-layer ANN model to advance image processing techniques for blurred or noisy images. They generated blurred images using low-pass spatial filtering and then recovered the filtered scales. Notably, their model validated the universal similarity hypothesis of Kolmogorov (1941), confirming that the two-dimensional and three-dimensional results exhibited homogeneous and isotropic turbulence. They compared probability density functions of real data with reconstructed data, demonstrating the effectiveness of their approach (Figure 25).

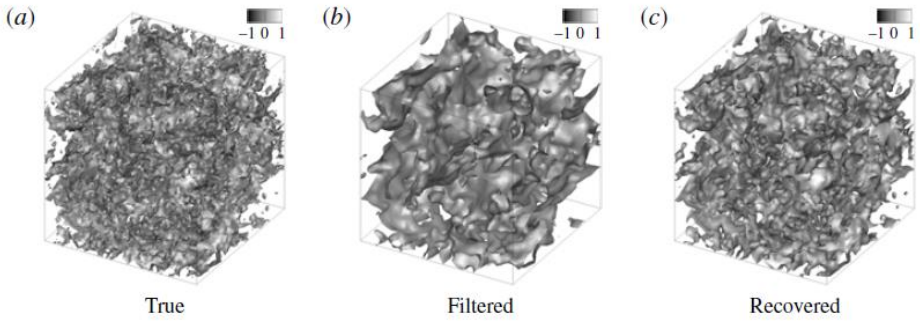

Figure 25. (a) A priori results for velocity field recovery for stratified turbulence true fields, (b) coarse-grained fields with Gaussian smoothing, (c) reconstructed fields (Maulik et al., 2017)

In the field of image processing, the application of unsupervised machine learning models has also been explored. Deng et al. (2019) developed two GAN models for reconstructing the images which had

low-resolution. These models can recover instantaneous flow field, statistical flow quantities, and spatial correlations. Figure 26 illustrates the GAN architecture, consisting of a generator and discriminator. Unlike typical GAN models that use random noise as input data, these models take low-resolution images as input. Similarly, Kim et al. (2021) utilized a cycle-GAN for a super resolution construction. In this study, the machine learning algorithm was trained to reconstruct filtered velocity fields and perform a conversion from LES data to DNS data. The goal was to generate a complete domain field from partial data. During the training process, the ground truth for comparison and evaluation was provided by the DNS data.

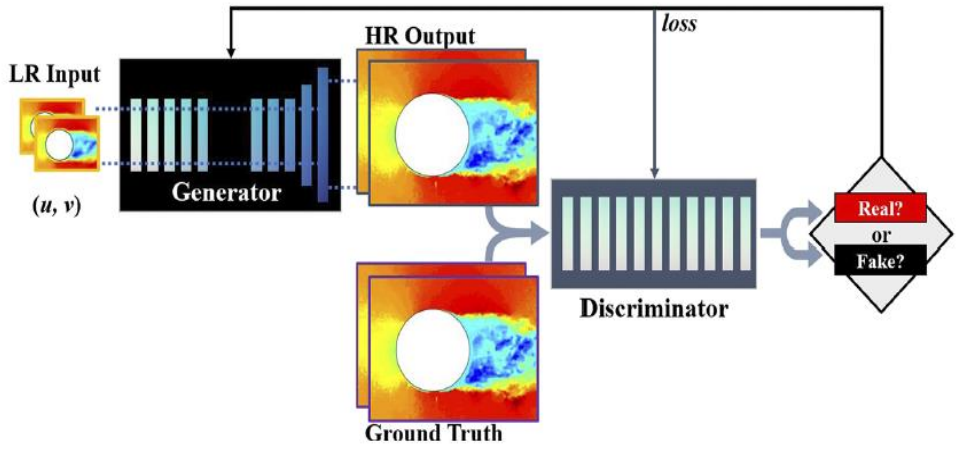

Figure 26. Schematic structure of super-resolution reconstruction of turbulent velocity fields using generative adversarial networks (Deng et al., 2019)

The second topic focus on solving turbulence models using neural networks (Ling et al., 2016; Wu et al., 2017; Gamahara & Hattori, 2017; Pawar et al., 2020; Park & Choi, 2021). Ling et al. (2016) proposed a neural network model that captures the non-linear relationship between Reynolds stresses and mean strain rate, unlike traditional linear eddy viscosity models used in Reynolds-Averaged Navier-Stokes (RANS) simulations. Wu et al. (2017) extended this idea by introducing both linear and non-linear functions to represent the Reynolds stress in their modified RANS model. Their approach improved the prediction of both Reynolds stress and mean velocity compared to the existing nonlinear

RANS model. Gamahara & Hattori (2017) focused on the relationship between grid scale and subgrid scale, deviating from the conventional assumption of subgrid-scale symmetry. They developed an artificial neural network (ANN) model that treated each subgrid-scale component separately, serving as a subgrid-scale model. Pawar et al. (2020) proposed a data-driven turbulent closure model, which estimated subgrid-scale stresses using resolved coarse flow fields. This data-driven approach exhibited computational efficiency compared to traditional subgrid-scale turbulent closure models. Park and Choi (2021) explored single and multiple neural network subgrid-scale models for predicting subgrid-scale (SGS) stresses in turbulent channel flow. They varied the input data, such as strain tensor and velocity gradient, and compared the results with Direct Numerical Simulation (DNS) data. Although this model had limitations in terms of geometry and applicability to specific Reynolds numbers, it demonstrated the potential of neural network models in capturing subgrid-scale phenomena.

The application of neural network models in solving non-linear PDEs has also been investigated. Raissi & Karniadaki (2018) developed a model to solve non-linear PDEs and assessed its accuracy by comparing the results with the actual solutions for various non-linear equations at different time points. They successfully applied the model to different PDEs, including Burgers' equation and Korteweg-de Vries (KdV) equation. Building upon this work, Raissi et al. (2020) extended the application of neural networks to solve the Navier-Stokes equation, a fundamental equation in fluid dynamics. They obtained the velocity and pressure fields by training neural networks on data from the Navier-Stokes equation. Additionally, they demonstrated the capability of neural networks in transforming low-resolution and noisy images into high-resolution images. This approach has been applied to various scenarios, including physical and biomedical problems. These studies show the potential of neural networks in solving non-linear PDEs and addressing hydrodynamic challenges. The use of ML techniques in hydrodynamics help improving understanding, prediction, and modeling in fluid dynamics applications. A summary of previous studies applying deep learning in hydrodynamics can be found in Table 5.

Table 5. Previous studies for hydro dynamic with deep learning

Papers	Methods	Classification	Purpose
Dong et al., (2015)	CNN	Image Processing	They proposed deep CNN model to recover single image super resolution with a little preprocessing and postprocessing. Especially they achieved superior results with the light computational cost. Also, the model had robustness and simplicity because they tested this model into various cases.
Ling et al. (2016)	DNN	Turbulent Model	They proposed a neural network model so that he Reynolds stresses and the mean strain rate have a nonlinear relationship, unlike the existing RANS model with a linear eddy viscosity model. The Reynolds stress of proposed model was more accurate by setting the DNS model result as the ground truth and comparing that of existing RANS model.
Wu et al. (2017)	ML	Turbulent Model	They modified the previous nonlinear model of RANS to divide the Reynolds stress into linear and nonlinear. The existing nonlinear model predicted the Reynolds stress well, but the mean velocity not. While, the proposed model solved the problem as predict the mean velocity well.
Maulik & San (2017)	ANN	Image Processing	They developed the ANN model to advance the image processing for blurred or noisy images. The ANN model had only a single layer and recovered the filtered scales. They confirmed the two-dimensional and three-dimensional results had homogeneous and isotropic turbulence.
Gamahara & Hattori (2017)	ANN	Turbulent Model	They found the relationship between the grid scale and subgrid-scale. They didn't focus on the symmetry of the subgrid-scale, but they considered each subgrid-scale component separately. The results confirm that ANN can be a substitute of the subgrid model.
Jin et al. (2018)	CNN	Image Processing	They developed a fusion CNN which had three-paths to capture spatial and temporal information. It predicted the time-series velocity fields from the pressure fields. The developed model can predict the velocity fields at various Reynolds numbers.

Raissi & Karniadaki (2018)	NN	Non-linear PDEs	They made a model to solve the non-linear partial differential equations (PDEs). The accuracy of the model was analyzed by comparing the results of the presented model with the actual solution for various non-linear equations at various time points.
Deng et al., (2019)	GAN	Image Processing	Two models were developed to reconstruct the images which had low-resolution. These models can recover instantaneous flow field, statistical flow quantities, and spatial correlations. They set the DNS data as ground truth and made the experimental results as high-resolution images.
Pawar et al. (2020)	ANN	Turbulent Model	The data-driven turbulent model was proposed and estimated the subgrid-scale stresses using resolved the coarse flow fields. It was called data-driven turbulent closure model and evaluated with DNS results. It had fast computational cost.
Raissi et al. (2020)	NN	Non-linear PDEs	They solved the non-linear partial differential equations (PDEs) and obtained the velocity and pressure fields. They also changed the low resolution and noisy images to high resolution. It was applied to the various situation such as physical and biomedical problems.
Park & Choi (2021)	ANN & CNN	Turbulent Model	They proposed some SGS models for predicting the SGS stresses for a turbulent channel flow. They changed the input data such as strain tensor and velocity gradient and compare the results with DNS data. This model has the limitation of the geometry.
Kim et al. (2021)	GAN	Image Processing	The cycle-GAN was developed for a super resolution construction. This model was used to reconstruct the filtered velocity fields and extracted the DNS data from the LES data. Also, it generated the full domain field from the partial data field.
GAN: Generative Adversarial Network; ML: Machine Learning; NN: Neural Network; ANN: Artificial NN; DNN: Deep NN; CNN: Convolutional NN			

3.3 Decomposition Method Literature Review

Several decomposition methods, such as Fast Fourier Transform (FFT), Proper Orthogonal Decomposition (POD), and Turbulence Filtering method, have been employed to analyze turbulence and extract flow structures. Among these methods, POD has been widely used in previous research papers (Chatterjee, 2000; Erdil et al., 2002; Oh et al., 2007; Bayraktar & Yilmaz, 2011; Karasu, 2020). Chatterjee (2000) focused on managing large amounts of data and aimed to obtain low-dimensional descriptions using POD that capture the significant flow structures. This decomposition method allows for the assessment of the energy contained in each mode, which is related to the kinetic energy of the fluid in fluid mechanics. Subsequently, Erdil et al. (2002) and Bayraktar & Yilmaz (2011) ranked the energy associated with each mode. In the study conducted by Karasu (2020), the diamond-shaped flow behind a cylinder was investigated, and the vortex shedding and shear layer were examined using decomposition methods, including POD. However, it is important to note that the POD method is sensitive to changes in coordinates (Chatterjee, 2000) and has limitations when applied to non-repetitive motions or high Reynolds number flows (Erdil et al., 2002).

The Fourier Transform (FT) method has also been widely used in various studies on turbulence analysis (Yılmaz & Kodal, 2000; Starn, 2001; Bayraktar & Yilmaz, 2011; Dubey, 2014; Karasu, 2020; Agarwal et al., 2021). Yılmaz & Kodal (2000) conducted experiments on a forced jet and applied FT to decompose the fluctuating and forced parts of the velocity fields. They identified the various frequencies present in the jet flows and specifically identified the frequency associated with the forced part. They highlighted that the FT method can effectively separate the forced part compared to the POD method. Starn (2001) utilized FT to efficiently solve fluid equations in large-scale simulations that require significant computational time. By transforming the images into the frequency domain, they were able to smooth the images and extract information corresponding to specific frequencies. This approach allowed for the reconstruction of images based on specific frequency components. Dubey (2014)

employed two-dimensional Fourier analysis to describe images. By converting the original images into the frequency domain, they were able to extract image features specific to certain frequencies. The two-dimensional FT method preserved the original image information while providing insights into frequency-related characteristics. Agarwal et al. (2021) performed numerical simulations using LES and employed FT to identify the modal structure of the flow. They effectively captured vortex features in specific areas using the FT method. Additionally, they utilized Dynamic Mode Decomposition (DMD), another decomposition method, to identify modes associated with vortex shedding and separation.

The turbulence filtering method has been utilized to extract proper coherent structures related to turbulence characteristics (Erdil et al., 2002). This method aims to separate the turbulent and non-turbulent parts of the flow by applying appropriate filters. Spatial filtering has also been employed to decompose velocity fields into turbulent and non-turbulent components (Oh et al., 2007). Unlike POD, which may not be suitable for decomposing turbulent flows, spatial filtering techniques allow for the extraction of turbulent eddy structures. It can be helpful for identifing and analyzing the dominant features and structures associated with turbulence. Bayraktar & Yilmaz (2011) conducted experiments using hot-wire anemometry and applied wavelet techniques. Wavelet analysis is effective in revealing the dominant frequencies present in the flow. By applying wavelet transforms, the multiscale characteristics of turbulence can be gained.

Table 6. Previous studies for decomposition method of velocity fields

Papers	Methods	Classification	Purpose
Yılmaz & Kodal (2000)	Turbulence filter method (FFT)	Decomposition turbulent and forced parts	They obtained the experimental results of a forced jet and decomposed the fluctuating part and forced part of velocity fields. They found out the various frequency of the jet flows and find the frequency of forced part. They found out that

			this method can take apart the forced part well compared with POD method.
Chatterjee (2000)	POD	Finding out the interesting flow structure	This study dealt with amount of data and tried to obtain the low-dimensional descriptions which can capture the interesting flow structures using POD. This decomposition method confirmed that how much energy each mode is contained, and these energies were related to fluid's kinetic energies in fluid mechanics area. However, POD method was sensitive to coordinates changed.
Starn (2001)	FFT	Solving the fluid equation easily	They used FFT to solve the fluid equations efficiently because the large fluid simulations took large cost time. They changed the image as frequency domains and made the images smoother using this frequency domain. Also, they redraw the image which was corresponding to specific frequencies.
Erdil et al. (2002)	POD & Turbulence filter method	Obtaining of various physical information	They verified that POD method can provide some modes which is ranked by the amount of energy, however it is hard to find out the coherent structure at high Reynolds number. Also, POD method needed identically repeated motions. While, the turbulence filter method can produce the proper coherent structure which is related turbulence characteristics.
Oh et al. (2007)	POD & Spatial Filter	Decomposition turbulent and non-turbulent parts	They conducted laboratory generated wind and decomposed the turbulent and non-turbulent parts. The turbulent coherent structure was identified by decomposition of the velocity fields. They confirmed that proper orthogonal decomposition was not good method to decompose the turbulent flows. While, the spatial filter can extract the turbulent eddy structure.
Bayraktar & Yilmaz (2011)	FFT, wavelet & POD	Comparison the decomposition technique	They experimented hot-wire anemometry and applied FFT, wavelet and POD technique into the results. They compared the free jet spectra and transverse jet spectra applied these decomposition techniques. They showed that the flow near the center of nozzle contained less energy. Also, POD results implied that the first

			few modes had significant amount of total energy.
Dubey (2014)	FT	Filtering two-dimensional images	A two-dimensional Fourier Transform was a good tool to describe the images. This image can be changed into frequency domain, so this method was used to extract the images for specific frequencies. This method also can preserve the information of original images.
Karasu (2020)	FFT & POD	Finding out the flow structure	In this experimental study, they investigated a diamond-shaped flow behind the cylinder. The effects a ratio of slit width over cylinder diameter was a parameter of this study. They can examine the vortex shedding and shear layer using the decomposition methods, FFT and POD.
Agarwal et al. (2021)	FFT & DMD	Verification the specific vortex structure using decomposition method	They obtained the numerical results using large eddy simulation and found out the modal structure using FFT and DMD. Also, they confirmed the vortex features in a specific area more clearly using FFT. And the modes related to specific vortex shedding and separation was indicated using DMD.

CHAPTER 4. METHODOLOGY

4.1 Numerical Model Descriptions

The simulation is executed on 40 nodes of the cluster, which is located in the Flow Physics and Informatics Laboratory at Seoul National University, and the parallel processes conducted by these cluster increase computing power and improve reliability. To perform the parallel process, Centos Linux, gcc, and MPICH, with versions 7.4, 4.94, and 3.1.5, are used for operation, compilation, and parallel processing, respectively. Table 7 provides a summary of the cluster server specifications. This study uses OpenFOAM, an open-source CFD program, to generate a turbulent jet. This software consists of C++ libraries and the application is developed with these libraries to generate executable files and conduct simulations (Greenshields, 2015). The application is divided into two categories: solvers and utilities. The solver refers to a fluid analysis program, while the utility refers to a program required for post-processing or format conversion. OpenFOAM gives users access to solvers and utilities, enabling them to easily solve problems by selecting the desired solvers and utilities. In addition, a folder is provided for pre- and post-processing, and the user can modify these two folders as desired. Figure 27 demonstrates the structure of OpenFOAM (Greenshields, 2015).

Table 7. Cluster specifications

R740 Server	Dell (TM) PowerEdge R740 Server
CPU	Xeon Gold 6248 20Core 2.5Ghz 27.50MB Cache × 2
RAM	12× 16GB DDR4 2933 Dual Ranked RDIMMs
Storage	43TB SSD server
OS	CentOS Linux version 7.5
Compiler	Gcc version 4.9.4
Parallel Computing	MPICH version 3.1.5
CFD Software	OpenFOAM-5.x with OpenMPI 2.1.1

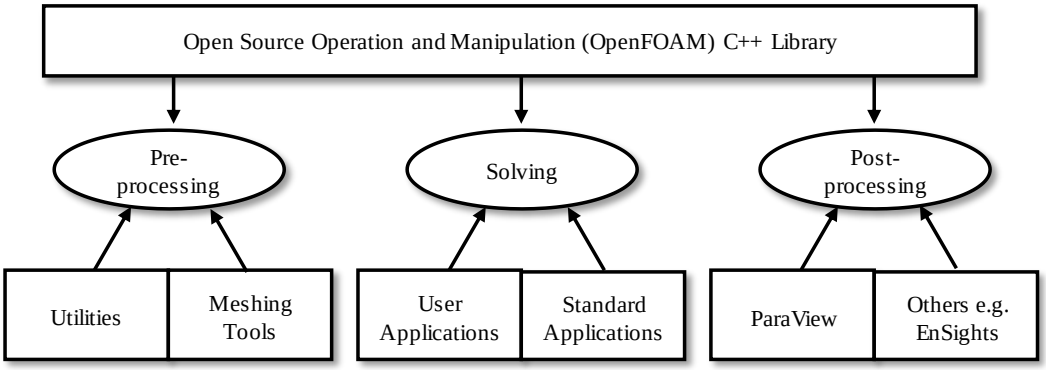

Figure 27. Structure of OpenFOAM (Greenshields, 2015)

This software uses numerical analysis technique, which is a method for obtaining a solution more quickly and easily by transforming a differential equation into an algebraic equation. Finite Difference Method (FDM) and Finite Volume Method (FVM) are examples of typical numerical analysis techniques. FDM permits each grid point to have a variable value, enabling a single algebraic equation for the variable value. The formula is resolved by using Taylor expansion to define the derivative. Since this finite difference method only has variables at lattice points, it is more beneficial for structural analysis than fluid flow studies. Therefore, OpenFOAM software employs the FVM to analyze fluid flow (Peiró & Sherwin, 2005). FVM is utilized to compute continuum mechanics associated with fluid flow and deformation (Greenshields, 2015). The finite volume method divides and analyzes the space to be analyzed into control volumes (CVs). Each volume is referred to as a cell. The control volume's surface consists of four in 2D and six in 3D. As the sum of the planes, the flux passing through the surface of the control volume is expressed. There is an illustration of the two-dimensional finite volume method, and because it is a two-dimensional finite volume method, there are four surfaces (S) e, w, n, and s, as depicted in Figure 28. Because the flux change in the cell equals the sum of the flux changes on each surface, the control volume method can be expressed as follows (Ferziger et al., 2002):

$$\int_S f dS = \sum_k \int_{S_k} f dS \tag{90}$$

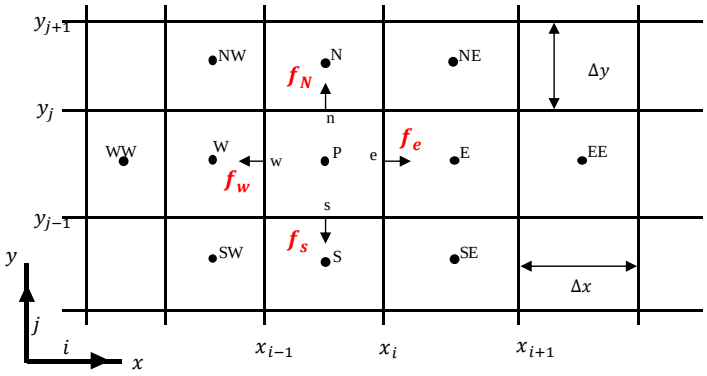

Figure 28. Example of 2D Finite Volume Method (Ferziger et al., 2002)

In this study, the Large Eddy Simulation (LES) equations were solved using the PIMPLE (PISO/SIMPLE) loop, which is a combination of the SIMPLE and PISO loops. The PIMPLE algorithm involves three main steps: (1) momentum prediction, (2) pressure correction, and (3) momentum correction. Initially, the velocity and pressure fields were estimated and inserted into the momentum equation to obtain a predicted velocity field. The guessed pressure field was then used to correct the predicted velocity field to satisfy the momentum equation. Subsequently, the corrected velocity field was used to rectify the pressure field, and the corrected pressure field was used to correct the velocity field once again to satisfy the momentum equation. The new velocity and pressure fields were then checked for convergence by substituting them into the continuity and momentum equations. If they converged, they were considered a solution, and the velocity and pressure fields for the next time step were estimated. If they did not converge, a new estimate was required at the same time step. The PIMPLE loop involves both inner and outer loops, with the outer loop beginning with the momentum predictor and the inner loop beginning with the pressure corrector. The PISO loop has one outer loop and several inner loops, while the SIMPLE loop has one inner loop and several outer loops. Since the PIMPLE loop is a combination of the PISO and SIMPLE loops, it has several inner and outer loops.

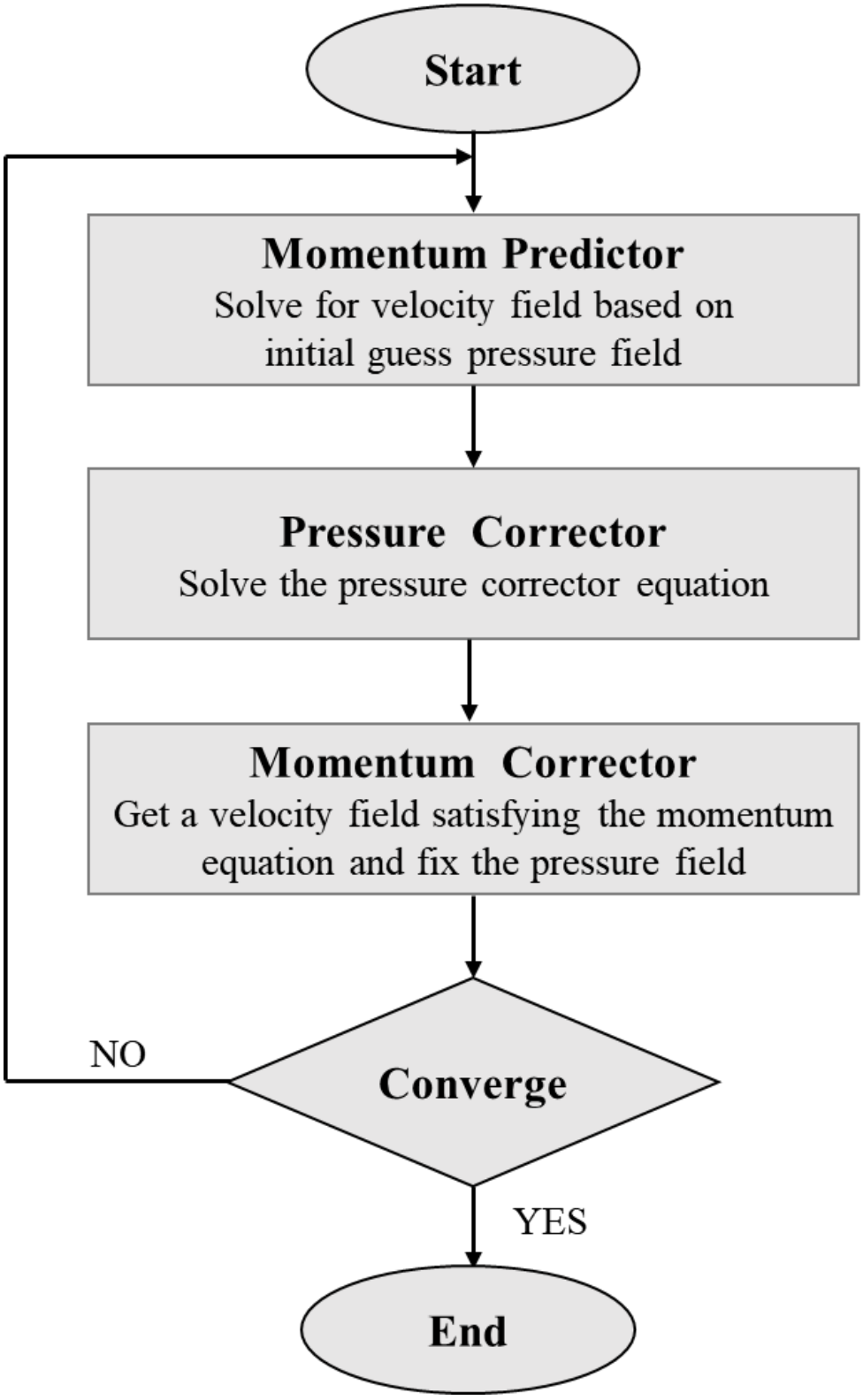

Figure 29. Pimple algorithm in OpenFOAM software.

4.2 Fourier Transform

This section introduces an application of the two-dimensional Fourier Transform to the jet flow velocity field. Figure 30a depicts a velocity field with an inlet velocity of $0.5\ m/s$ and a diameter of $0.02\ m$. The velocity field is the fluctuating velocity obtained by subtracting the instantaneous velocity from the mean velocity. The mean velocity is computed by averaging the instantaneous values from 350 to 450 seconds after the jet inlet emission. In stream-wise and span-wise directions, the domain range is 0 to 0.8 m and the uniform grid size is 0.05 m. The total number of grids is 160 × 160 in x and y. The velocity range is between 0 and 0.18 m/s, and the velocity fluctuation is clearly visible in the shear layer. Figure 30 shows the velocity field and its centered log-scale Fourier Spectrum. The centered logarithm Fourier Spectrum has ranges from 0 to 6, with high energy at low frequency.

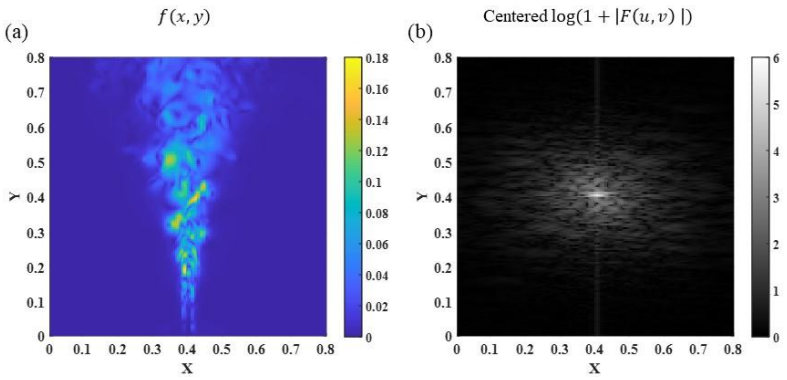

Figure 30. (a) The velocity field and (b) its Fourier Spectrum

It is conducted as depicted in Figure 31 to check the effects of high pass and low pass filtering on the original jet velocity field. As mentioned previously, the original image of a jet is the magnitude of fluctuating velocity (Figure 30a). The component, h_0, is set to 15 in Equation 58 and 59. The low pass filter retains the desired low frequencies while setting the Fourier values of undesirable high frequencies to 0. Since the high Fourier spectrum values are concentrated at low frequencies, the jet flow has an

abundance of energy at low frequencies. So, the low filtering retains a significant amount of energy, and the maximum velocity is 0.18 m/s, which is the same value as the original image (Figure 31a). It is also confirmed that the jet flow is reproduced in its entirety. Figure 31b shows that the low pass filtering makes the image smoothing. The spatial energy spectra cut off at the certain value which is set by the component, h_0, in Equation 58 and 59. The spatial spectrum reveals that the frequency is proportional to the energy content, with the lower the frequency containing more energy (Figure 31c). In addition, if the energy spectrum is examined after the low pass frequency has been applied, it notices that the energy is cut off at a specific frequency. Using this, the part of velocity in grid smaller than the sub grid scale can be eliminated.

While, the image's high frequency components are preserved when applying the high pass filter (Figure 31d). Small movements are captured when using the high pass filtering, unlike with low pass filtering. The maximum velocity is 0.09 m/s, which is approximately half of the original image's maximum velocity. In addition, high pass filtering is used to express sharp movements. Also, the high pass filtering makes the Fourier spectrum value of other frequencies zero except for the specific high frequencies (Figure 31e). The energy spectrum of high pass filtering is only illustrated at high frequencies (Figure 31f). It has been demonstrated that when the frequency is high, the energy is relatively low. When the energy spectrums of high and low filtering are combined, the result is the original energy spectrum of the image.

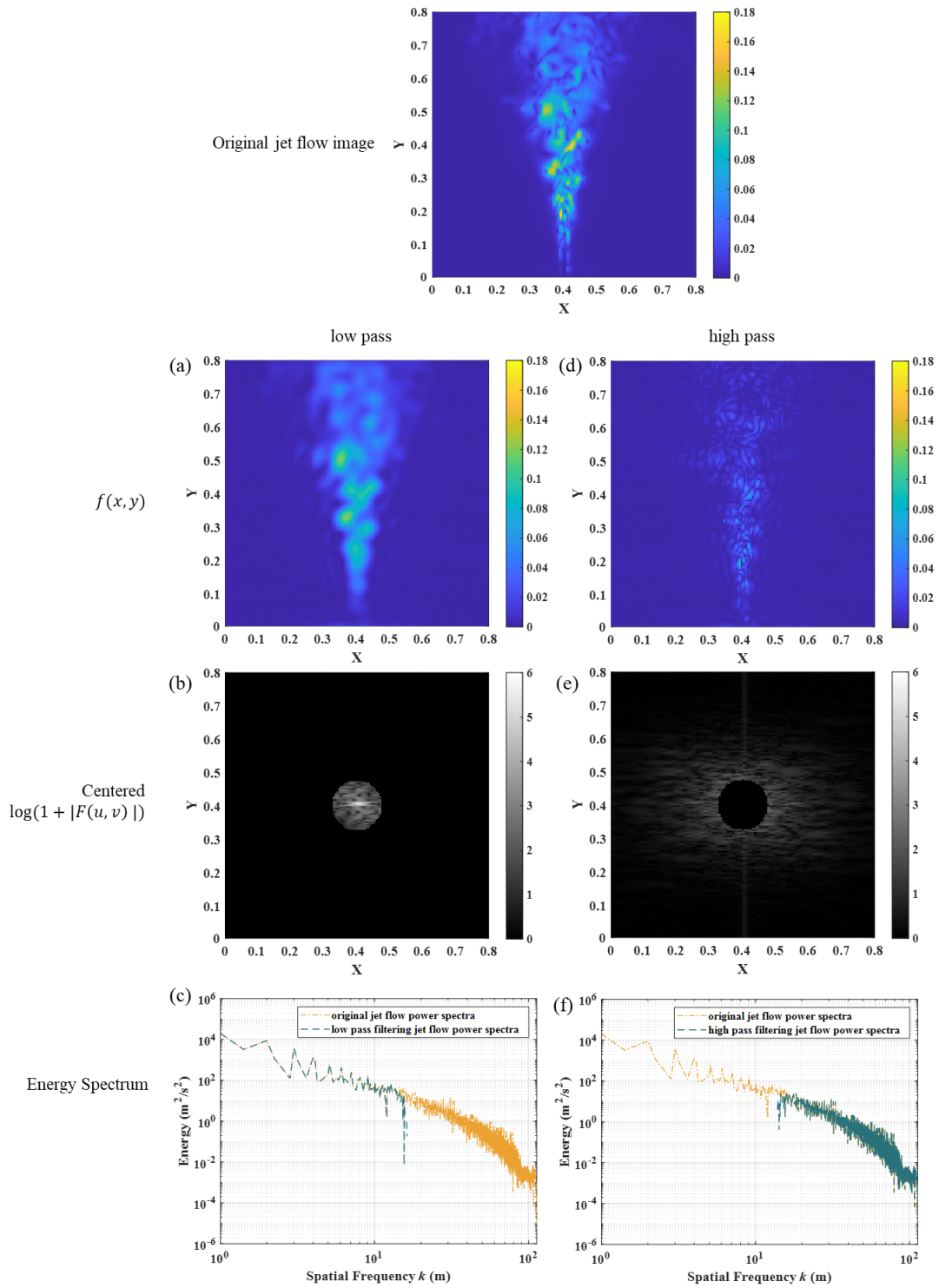

Figure 31. (a) The high pass filter velocity field image, (b) its Fourier Spectrum, (c) its spatial Power Spectra and (d) low pass filter velocity field image, (e) its Fourier Spectrum, (f) its spatial Power Spectra

4.3 Machine Learning Algorithms

4.3.1 Convolutional Neural Network Algorithm

This study examines two types of CNN algorithms to extract the jet flow image and predict the jet flow at the next time step. The first algorithm is a conventional model comparable to the one proposed by Lecun (1989). This model has a single path composed of an input layer, a convolutional layer, a pooling layer, and an output layer (see Figure 32a). This CNN model with a single path contains six convolutional blocks with convolutional layers, nonlinear functions, and max pooling layers. The convolutional block right before the output layer consists of simply a convolutional layer. Additionally, the model includes an LSTM model, which is one of the RNN models. It helps to remember the time-series data. The nonlinear function employs LeakyReLU, which can output both positive and negative integers without diverging in the negative direction. The second CNN model is the model suggested by Jin et al. (2018), which combines three paths, as illustrated in Figure 32b. This model is intended to capture both precise spatial-temporal information and traits that are invariant to tiny spatial-temporal series variations (Jin et al., 2018). This model consists of six convolutional blocks, each comprising a convolutional layer, nonlinear function, maximum pooling layer, and fully connected layer. Additionally, this model includes an LSTM model at the beginning of the second CNN model. This model sends data over three paths and turns image features into one-dimensional data using a fully connected layer after passing through the convolutional layer. Concat layer captures and combines certain image characteristics. And the final block reproduces the image with these characteristics. The second CNN model uses a nonlinear function as LeakyReLU for the same reason as the first CNN model. In both models, the optimizer was trained with Adam, which can take gradient direction and learning rate into account. In this study, two CNN models are evaluated to identify which model reproduces the highest resolution LES velocity fields more accurately, and then the suitable range by adjusting the grid size and Reynolds number are indicated.

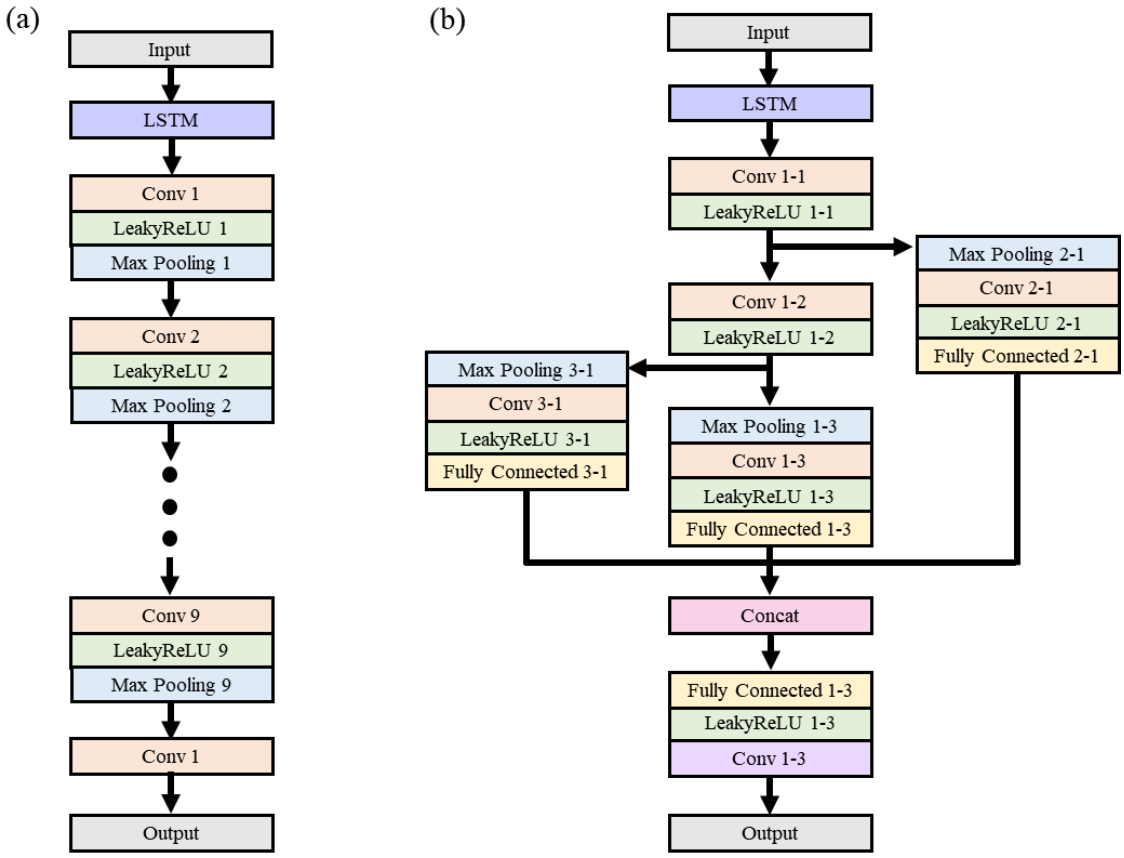

Figure 32. (a) The traditional CNN architecture and (b) the three path CNN architecture; 'Conv' and 'Concat' denote convolution layer and Concatenate

In the CNN algorithm, the optimal kernel size, batch size, and epoch settings are crucial time-saving and performance-enhancing parameters. So, in this study, the proper hyperparameters of the second model are investigated. First, to determine the appropriate batch kernel size, the time and epoch required when using various kernel sizes are investigated. Figure 33 shows the train and validation loss as a function of kernel size, with the red circle representing the optimal epoch for each case. All cases train the algorithm with batch size 2. In all cases, the train loss is decreasing, whereas the validation loss remains constant before increasing abruptly. The optimal epoch is when the validation loss exceeds the training loss. As the kernel size increases, the number of training epochs increases, but the loss does not change significantly. Table 8 summarizes the optimal epoch and total training time for each kernel size.

As the kernel size increases, the total training time until 100 iterations take longer (Table 8). Therefore, the optimal kernel size is 3×3 because it consumes less time and requires fewer epochs.

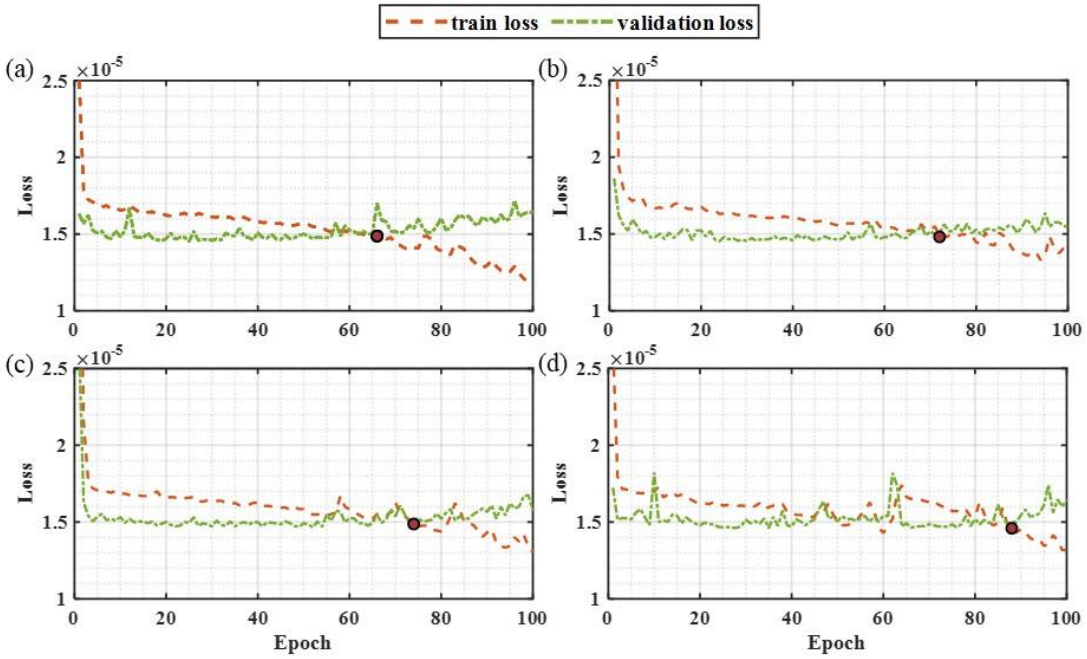

Figure 33. Train and validation loss of kernel size (a) 3×3, (b) 5×5, (c) 7×7 and (d) 9×9 and the optimal epoch for each case marked by a circle filled red

Table 8. Optimized epoch and total training time until 100 epochs for kernel size 3×3, 5×5, 7×7 and 9×9

kernel size	3×3	5×5	7×7	9×9
Optimized epoch	66	72	74	88
Total Time	1265.6 seconds	1413.4 seconds	1635.2 seconds	1834.2 seconds

Figure 34 depicts the train and validation loss depending on batch, and the red color circle shows the optimal epoch for each case. All cases train the algorithm with kernel size 3×3. The optimal epoch is usually set the validation loss is higher than the training loss. In cases with batch sizes 1, 2, and 5, the train loss decreases while the validation loss remains constant before increasing abruptly. It is confirmed

that the sudden increase as occurring immediately after the optimal epoch. When the batch size is 10, however, there is no proper epoch because train loss exceeds validation loss in every epoch. It indicates that batch 10 requires more than100 epochs. The optimal epoch and total training time for each kernel size are summarized in Table 9. This table demonstrates that as batch size increases, total training time decreases, but more epoch are required to reach proper training. But, the loss did not differ significantly between cases. The batch size of five spends less time calculating the time until the appropriate epoch is reached. So, this study set the batch size to 5.

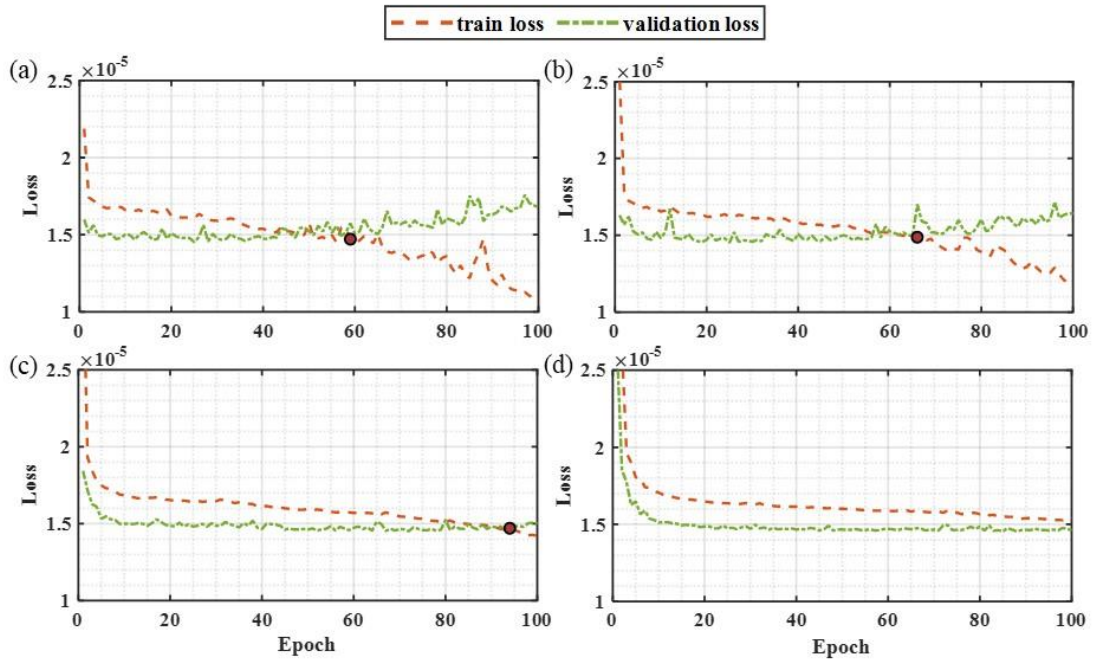

Figure 34. Train and validation loss of batch size 1, 2, 5 and 10 and the optimal epoch for each case marked by a circle filled red

Table 9. Optimized epoch and total training time until 100 epochs for batch size 1, 2, 5 and 10

Batch size	1	2	5	10
Optimized epoch	59	66	94	-
Total Time	2250.4 seconds	1265.6 seconds	675.6 seconds	537.6 seconds

Table 10. Details of the parameters of the second CNN structure

Type	Layer	Kernel size/stride	Activation	Output shape
Main path	Input	-	-	120 × 120 × 3
	LSTM	3 × 3/2	Leaky ReLU	120 × 120 × 3
	Conv 1-1	3 × 3/2	Leaky ReLU	60 × 60 × 8
	Conv 1-2	3 × 3/2	Leaky ReLU	30 × 30 × 16
	Max Pooling 1-3	3 × 3/1	-	30 × 30 × 16
	Conv 1-3	3 × 3/2	Leaky ReLU	15 × 15 × 32
	Flatten 1-3	-	-	7200
	Dense 1-3	-	-	300
	Concatenate	-	-	900
	Dense 1-4	-	Leaky ReLU	10800
	Reshape	-	-	60 × 60 × 3
	ConvTranspose 1-4	3 × 3/2	-	120 × 120 × 3
	Output	-	-	120 × 120 × 3
Right path	Max Pooling 2-1	3 × 3/2	-	30 × 30 × 16
	Conv 2-1	3 × 3/2	Leaky ReLU	15 × 15 × 32
	Flatten 2-1	-	-	7200
	Dense 2-1	-	-	300
Left path	Max Pooling 3-1	3 × 3/1	-	15 × 15 × 32
	Conv 3-1	3 × 3/2	Leaky ReLU	15 × 15 × 32
	Flatten 3-1	-	-	7200
	Dense 3-1	-	-	300

Table 10 summarizes the second CNN model, which can be divided into three paths. As stated above, it consists of six convolutional boxes, which include convolutional layer, nonlinear function, maximum pooling layer, and fully connected layer. The first path is the main path that divides and gathers data. To keep the image's characteristics, there are left and right paths. To be specific, after the LES data sets are obtained, the highest resolution LES data set is used as truth data known as ground truth, while the LES data set is used as training set and test set. In other words, the highest resolution LES data is used to determine train the sub-grid eddies and evaluate whether these eddies are reproduced the sub-grid scale of highest resolution LES data. It is specifically trained to replicate not only velocity fields but also energy spectra. The sub-grid length scale is then determined using the wave number, which is a

spatial frequency of the eddy. In this study, the Fourier Transform was used to decompose the sub-grid scale and the resolved scale.

4.3.2 Generative Adversarial Network Algorithms

The standard GAN consists of two networks: the generator and discriminator. The generator produces the fake images, while the discriminator identifies them. The two GAN algorithms presented in this work also have a generator and discriminator. The first GAN algorithm is a traditional model consisting of Fully Connected (FC) block, Convolutional layer, and maximum pooling layer. FC Blocks contain a flatten layer, a dense layer, and a non-linear function (Figure 35a). The first GAN model constructed using FC blocks needs lots of time since all the data must be processed in each layer. The generator has six FC blocks and one CB, and the discriminator has ten FC blocks and one CB. The generator must produce a fake image, the output size of the final layer, convolutional layer, is same to the wanted image size. The final layer of the discriminator utilizes a sigmoid that returns a value between 0 and 1, since the image should return a value near to 1 if it is real data and close to 0 if it is fake data. This traditional GAN architecture is inadequate for capturing minor motions or statistical fluctuations. So, Deng et al. (2019) propose a super resolution GAN that is applied to the turbulent flows. This model is composed of CBs, fully connected layer, and upsampling layers (Figure 35b). The generator has six CBs, one fully connected layer, and two upsampling layers. Two upsampling layers are employed to make the output size match the intended image size. The discriminator's final layer is a sigmoid layer for the same reason as the first GAN method.

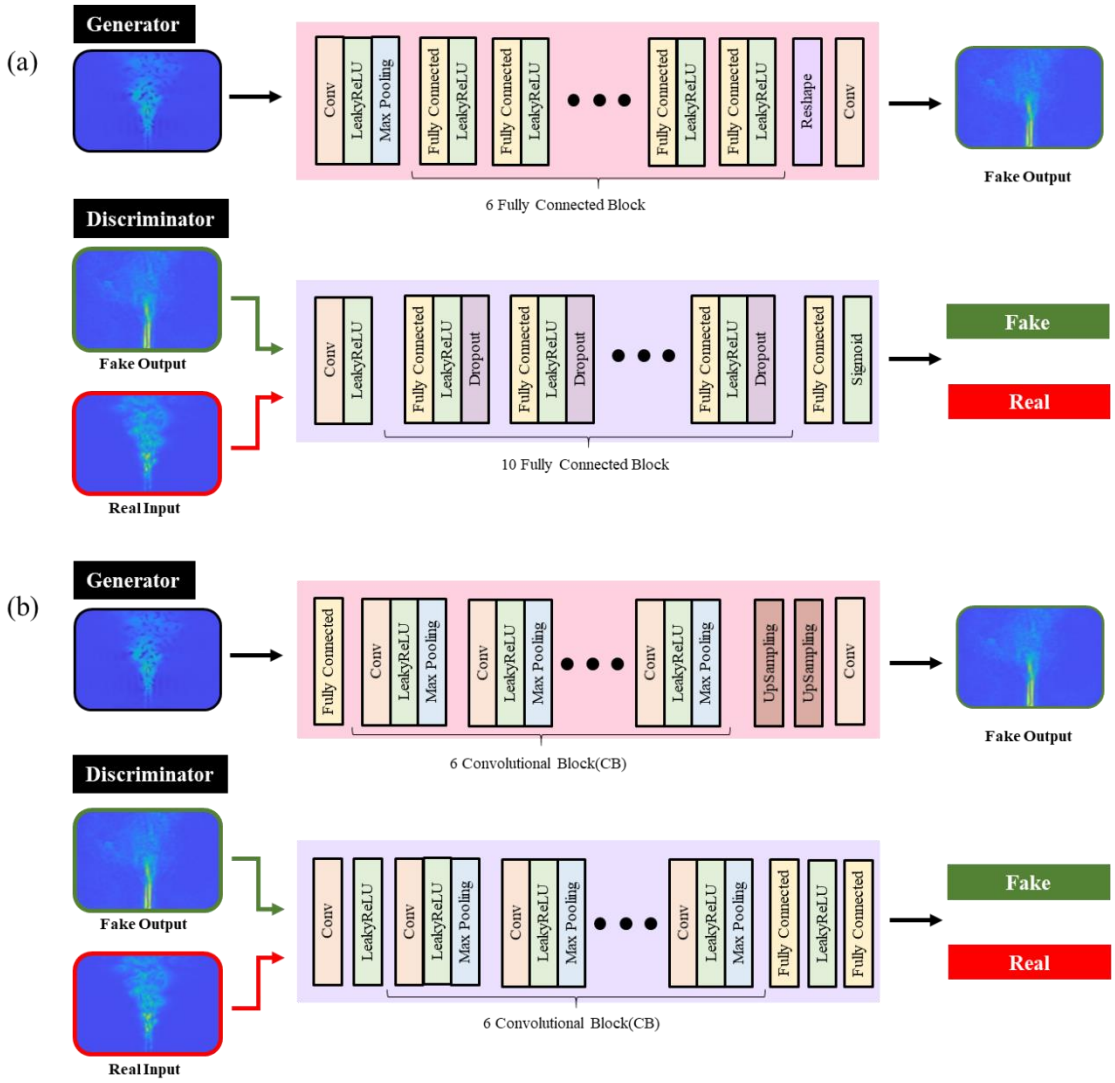

Figure 35. (a) The traditional GAN architecture and (b) the super resolution GAN architecture; 'Conv' denotes convolution layer

The loss function is one of the factors that has a significant impact on the performance of machine learning when determining the optimal weight parameters. In forward propagation, the difference between neural network output and real data is calculated, and the loss value is used to determine the optimal weights (Ho & Wookey, 2019). The loss funciton of GANs algorithms is the binary cross-entropy (log loss) as loss function. The binary cross-entropy is defined as (Jadon, 2020):

$$L(i,\hat{\imath}) = -\sum_{ND}(ilog(\hat{\imath}) + (1-i)log\ (1-\hat{\imath})) \quad (91)$$

where i and $\hat{\imath}$ are true values and the predicted results by GANs, respectively, and *ND* is the number of data sets. It compares predicted probability distribution to ground truth probability distribution, and it is usually used for a classification problem in machine learning (Hurtik et al., 2022). When this loss function is applied, the weight and bias parameters are modified so that the sum of loss value approaches 0. The loss value can only be zero or one. In other words, the loss is zero if the predicted value is categorized in real data, and one if it is not. The loss values corresponding to real data group in the loss function are as follows:

$$L(i,\hat{\imath}) = -\sum_{ND} i log(\hat{\imath}). \tag{92}$$

On the other hand, the loss value corresponding to the fake group is as follows:

$$L(i,\hat{\imath}) = -\sum_{ND}(1-i) log\ (1-\hat{\imath}). \tag{93}$$

The optimization in both GAN algorithm is Adam which is used in CNN algorithms.

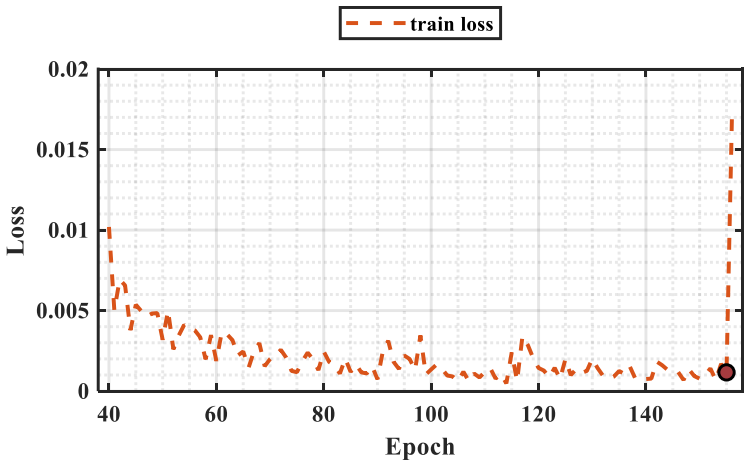

Figure 36. Train loss of second GAN algorithm and the optimal epoch for each case marked by a circle filled red

Table 11. Details of the parameters in the second GAN structure

Type	Layer	Kernel size/stride	Activation	Output shape
Generator	Input	-	-	120 × 120 × 3
	Conv	3 × 3/2	LeakyReLU	60 × 60 × 8
	Conv	3 × 3/1	LeakyReLU	60 × 60 × 16
	Max Pooling	3 × 3/1	-	60 × 60 × 16
	Conv	3 × 3/1	LeakyReLU	60 × 60 × 32
	Max Pooling	3 × 3/1	-	60 × 60 × 32
	Conv	3 × 3/1	LeakyReLU	60 × 60 × 64
	Max Pooling	3 × 3/1	-	60 × 60 × 64
	Conv	3 × 3/1	LeakyReLU	60 × 60 × 128
	Max Pooling	3 × 3/1	-	60 × 60 × 128
	Conv	3 × 3/1	LeakyReLU	60 × 60 × 256
	Max Pooling	3 × 3/1	-	60 × 60 × 256
	Conv	3 × 3/2	LeakyReLU	30 × 30 × 512
	Max Pooling	3 × 3/1	-	30 × 30 × 512
	UpSampling	3 × 3/2	-	60 × 60 × 64
	UpSampling	3 × 3/2	-	120 × 120 × 8
	Conv 2	3 × 3/1	LeakyReLU	120 × 120 × 3
	Output	-	-	120 × 120 × 3
Discriminator	Input	-	-	120 × 120 × 3
	Conv	3 × 3/2	LeakyReLU	60 × 60 × 8
	Conv	3 × 3/1	LeakyReLU	60 × 60 × 16
	Max Pooling	3 × 3/1	-	60 × 60 × 16
	Conv	3 × 3/2	LeakyReLU	30 × 30 × 32
	Max Pooling	3 × 3/1	-	30 × 30 × 32
	Conv	3 × 3/2	LeakyReLU	15 × 15 × 64
	Max Pooling	3 × 3/1	-	15 × 15 × 64
	Conv	3 × 3/2	LeakyReLU	8 × 8 × 128
	Max Pooling	3 × 3/1	-	8 × 8 × 128
	Conv	3 × 3/2	LeakyReLU	4 × 4 × 256
	Max Pooling	3 × 3/1	-	4 × 4 × 256
	Conv	3 × 3/2	LeakyReLU	2 × 2 × 512
	Max Pooling	3 × 3/1	-	2 × 2 × 512
	Fully Connected	-	LeakyReLU	512
	Fully Connected	-	-	1
	Output	-	-	120 × 120 × 3

To determine a proper training epoch of second GAN model, the loss per epoch is examined (Figure 36). It is confirmed that the loss value decreases progressively as the epoch grows and increases abruptly at epoch 150. So, the epoch and the batch size are set to 150, 10, respectively. The details of second GAN algorithm are summarized in Table 11.

CHAPTER 5. RESULTS

5.1 Computational Results of LES

The computational domain was chosen to encompass the jet flow, as depicted in Figure 37a. The domain extends up to $x = 0.6$m in the streamwise direction and 0.2 m in the cross-stream directions (r). Hefny and Ooka (2009) suggested that the choice of computational mesh can influence the computational results. They confirmed that the orthogonal hexahedral mesh style offered the most efficient computational solution compared to other mesh shapes. Other mesh styles produced poor results due to their skewness and misalignment with the predominant direction of flow. Therefore, this study employs regular hexahedral uniform mesh grids with four different resolutions, consisting of 1,008,000 to 2,560,000 control volumes. Additionally, the inlet velocity varies depending on the different Reynolds numbers considered. The velocity magnitude of the jet simulated using the LES turbulence model is depicted in Figure 37b. The figure illustrates the presence of a jet core and a shear layer, wherein the core thickness decreases as it moves away from the inlet. The shear layer is situated farther from the inlet compared to the core and exhibits a significant increase in thickness just above the jet inlet. The velocity magnitude field of the jet core appears to be higher than that of the shear layer. Table 12 presents information on the different mesh resolutions and flow conditions at the inlet for a total of five cases. It also summarizes the elapsed time required to achieve a simulation time of 2000 seconds for each case. It is observed that smaller grid sizes necessitate more time for computation. The dataset used to train the ML algorithms was obtained ten minutes after the simulation initiation. Data was saved every second for a duration of twenty minutes, with the final eight minutes utilized for algorithm evaluation. All of the cases in this study employ the LES turbulence model.

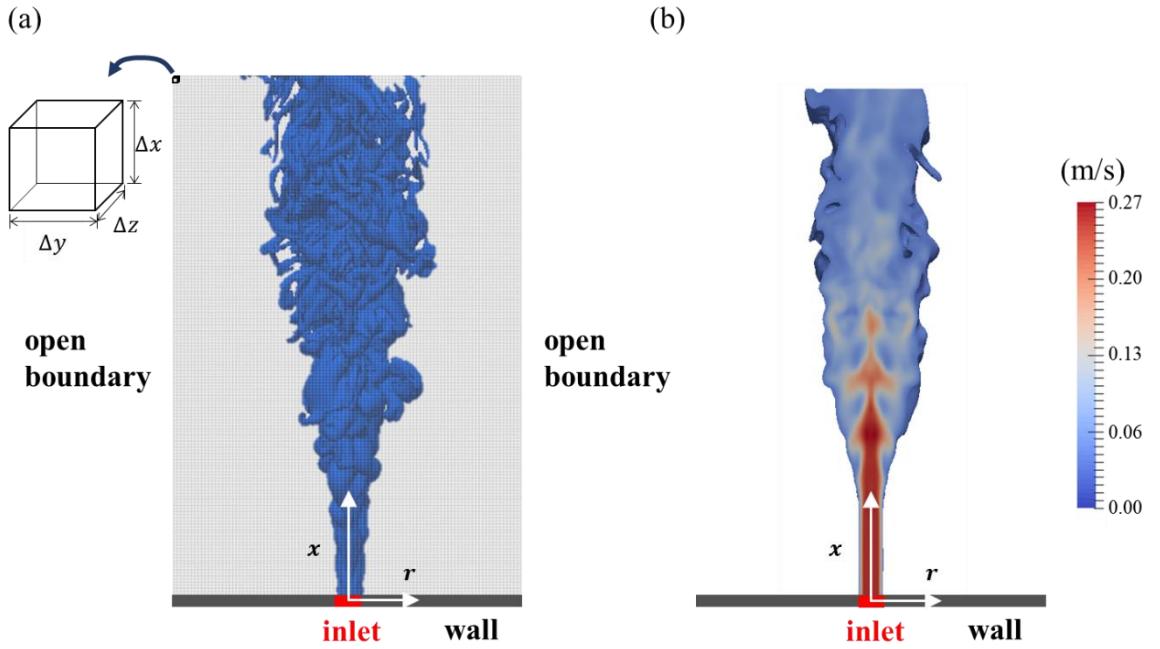

Figure 37. (a) The computational domain for the jet flow is shown with a blue-colored iso-surface of vorticity magnitude (10/s), (b) An axial section of the velocity field at a single instance is also displayed.

Table 12. Details of the computational cases.

Case	1	2	3	4	5
Grid resolution	161×161×101	141×141×101	121×121×71		
Cells (approximate)	2,560,000	1,960,000	1,008,000		
Grid spacing	0.5 cm	0.6 cm	0.7 cm		
Inlet velocity	0.5 m/s			0.75 m/s	1.0 m/s
Reynolds number	10,000			15,000	20,000
Elapsed time	1,536,000 seconds	1,124,000 seconds	514,000 seconds	530,000 seconds	544,000 seconds

The second numerical experimental case is conducted with a grid size of approximately 0.1 cm and considers Reynolds numbers of 10,000 and 15,000. The computational domain for this case has dimensions of 0.6 m in the axial direction and 0.2 m in the radial direction. The diameter of the jet is 2 cm, resulting in jet inlet velocities of 0.5 m/s and 0.75 m/s for the Reynolds numbers of 10,000 and 15,000, respectively. Table 13 provides a summary of the turbulent characteristics for the Reynolds number of 10,000 in Case 1 of Table 12. The corresponding equations for the turbulent characteristics are as follows:

$$k = \frac{1}{2}(u'^2 + v'^2 + w'^2),\ \varepsilon = 2\nu\langle s'_{ij}s'_{ij}\rangle + 2\nu_t\langle s'_{ij}s'_{ij}\rangle \tag{94}$$

where u', v', w', ν, and s'_{ij} denote the velocity fluctuations in the stream-wise, lateral, and vertical directions, kinematic viscosity, and strain rate tensor, respectively. The indices i=1, 2, and 3 represent the stream-wise, spanwise, and vertical directions, respectively. The other turbulence statistics are estimated using spatially averaged k and ε. The Taylor microscale (λ) and the associated Reynolds number (Re_λ) are defined as:

$$\lambda = \sqrt{15\nu q^2/\varepsilon},\ Re_\lambda \cong q\lambda/\nu \tag{95}$$

Table 13. Spatial averaged turbulence statistics

k (m^2/s^2)	ε (m^2/s^3)	λ (m)	Re_λ	τ_η (s)	η (m)
0.00075	4.95×10^{-5}	0.0212	813	0.142	3.77×10^{-4}

To determine the appropriate grid size for LES, four specific locations were chosen to calculate the characteristic length in the (x, r_1, r_2) directions. These locations, along with a schematic representation of the jet flow computational domain, are summarized in Figure 38 and Table 14. The autocorrelations at these four locations, at four different times, and in three directions are depicted in Figure 39-Figure

42. The point at which the autocorrelation first reaches zero is indicated with a marker. For the first probe, the autocorrelation becomes zero at approximately 0.0575 m. For the 2nd, 3rd, and 4th probes, the autocorrelation become zero at distances of 0.0660, 0.080, and 0.0865 m, respectively. Consequently, the characteristic length in this study is determined to be 0.0725 m.

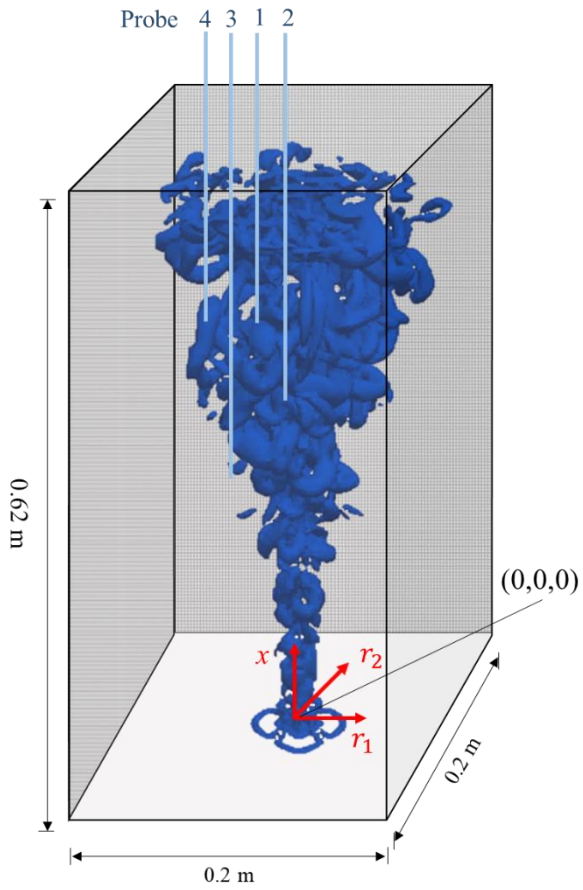

Figure 38. A schematic of the jet flow computational domain and indicates the four locations.

Table 14. The four locations used for calculating the characteristic length at probes.

Probe number	$(x,\ r_1,\ r_2)$
Probe 1	(0.603, -0.014, -0.190)
Probe 2	(0.517, -0.006, -0.022)
Probe 3	(0.430, -0.027, -0.027)
Probe 4	(0.603, -0.034, -0.039)

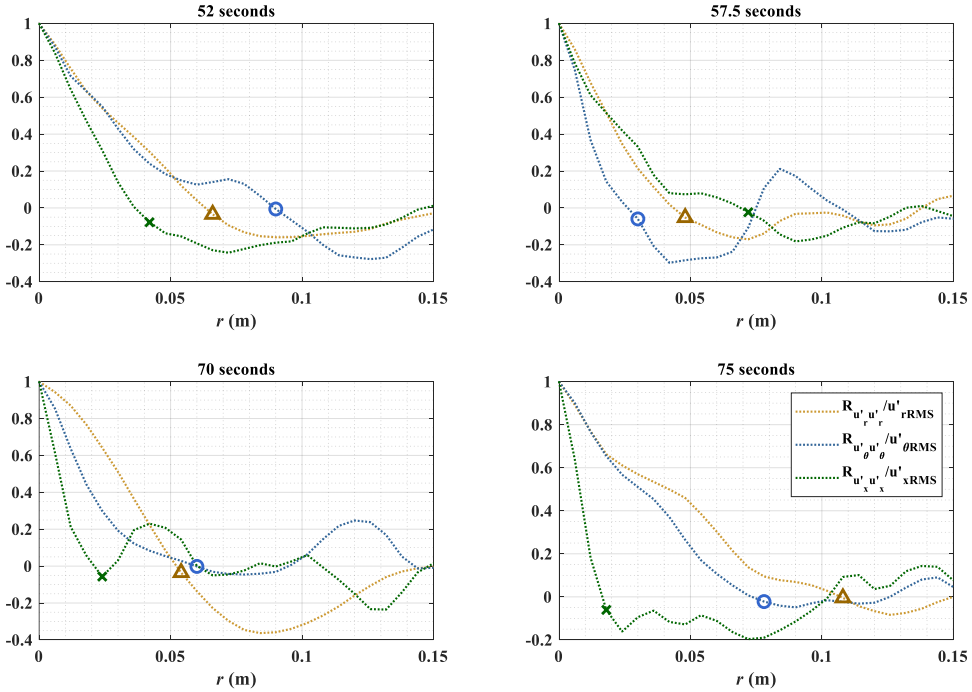

Figure 39. The autocorrelation of velocity in the streamwise, spanwise, and transverse directions was calculated for probe 1.

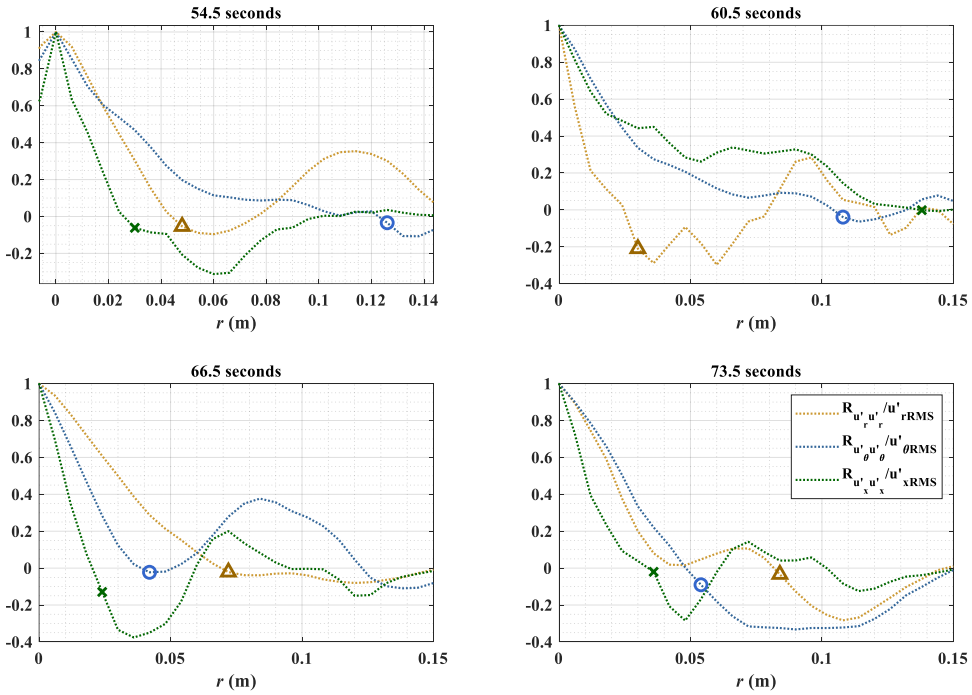

Figure 40. The autocorrelation of velocity in the streamwise, spanwise, and transverse directions was calculated for probe 2.

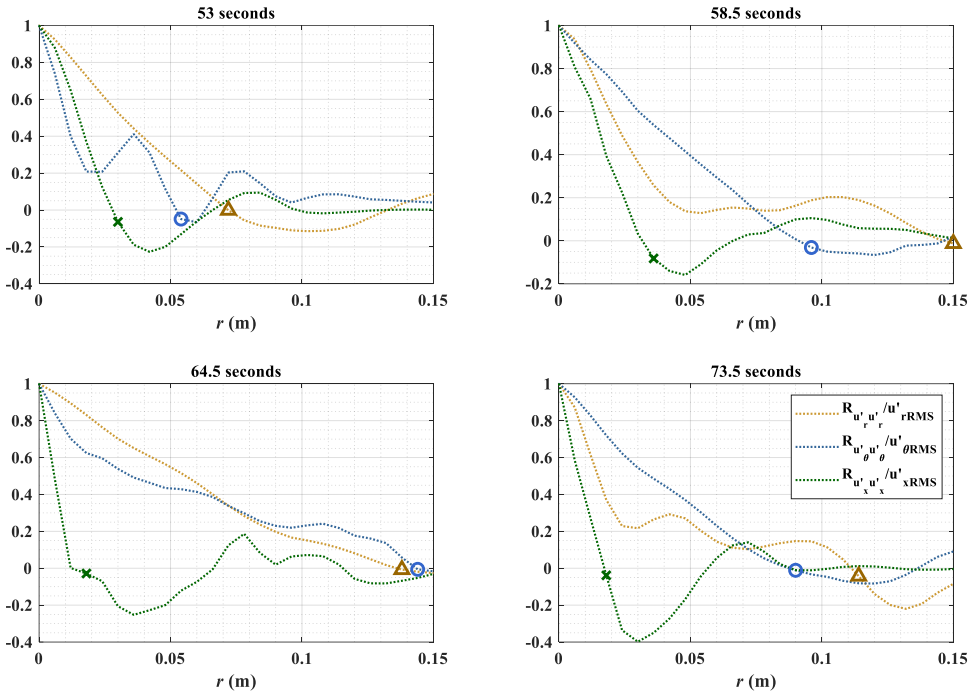

Figure 41. The autocorrelation of velocity in the streamwise, spanwise, and transverse directions was calculated for probe 3.

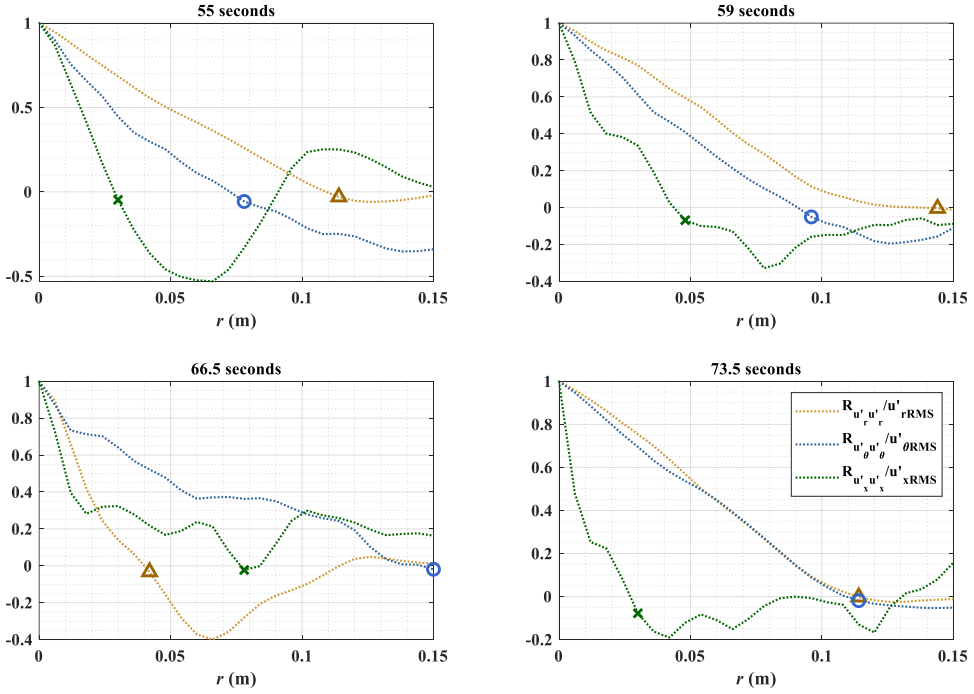

Figure 42. The autocorrelation of velocity in the streamwise, spanwise, and transverse directions was calculated for probe 4.

In the second study, the grid size employed in the domain was approximately 0.1 cm that is 60-70 times smaller than the characteristic length.

Four experimental results were used to validate the ground truth data in this study (Panchapakesan et al., 1993; Khorsandi et al., 2013; Moeini et al., 2020; Fang et al., 2020). Khorsandi et al. (2013) and Moeini et al. (2020) both used acoustic Doppler velocimetry (ADV) to examine jet flows with Reynolds numbers of 10,600 and 10,000, respectively, based on jet diameter and nozzle exit velocity. The sampling frequencies were 25 Hz and 200 Hz, respectively. Panchapakesan et al. (1993) used flying hot-wire anemometry (FHWA) to examine the jet flow at a Reynolds number of 11,000. Table 15 summarizes the comparison of the experimental results mentioned above with the present results. As shown in the table, the mean velocity decay constant of this study has the lowest value compared to those of the other experiments. Moreover, the mean velocity decay constant appeared to increase with increasing Reynolds number. Given that the Reynolds number in this study is the smallest, it is reasonable to conclude that the smallest value for the mean velocity decay constant is appropriate.

Table 15. Comparing the present results with three previous experimental studies conducted by Moeini et al. (2020), Khorsandi et al. (2013), and Panchapakesan et al. (1993) for validation purposes.

	Present work	Moeini et al. (2020)	Khorsandi et al. (2013)	Panchapakesan et al. (1993)
Method	LES	ADV	ADV	FHWA
Re	10,000	10,000	10,600	11,000
Jet diameter (m)	0.02	0.01	0.03175	0.0016
Mean velocity decay constant	5.05	5.58	5.66	6.06

Figure 43 depicts the profile of axial and lateral mean velocity, normalized by the nozzle diameter

(D) and the jet exit velocity (U_j), at a certain distance from the jet entrance, x_c/D=37. The results of this study are obtained at 100Hz frequency for 300 seconds, and then temporally averaged. The red dots in Figure 43a represent the locations where the velocity data were collected. Consistent with the finding of physical experiments, the axial velocity was shown to decrease along the r-axis axis as it moves away from the jet inlet (Figure 43b). In the meantime, some discrepancies are observed between the lateral mean velocity and that of other studies. However, Figure 43c shows the same trend between the simulated and observation data that the highest velocity is located at a similar x position, and then the velocity gradually slows down similarly. Especially, in line with the ADV experimental result of Moeini et al. (2020), it is confirmed that the velocity slightly increases at r/x_c of 0.15. Therefore, it is evident that the current LES results closely resemble the experimental results.

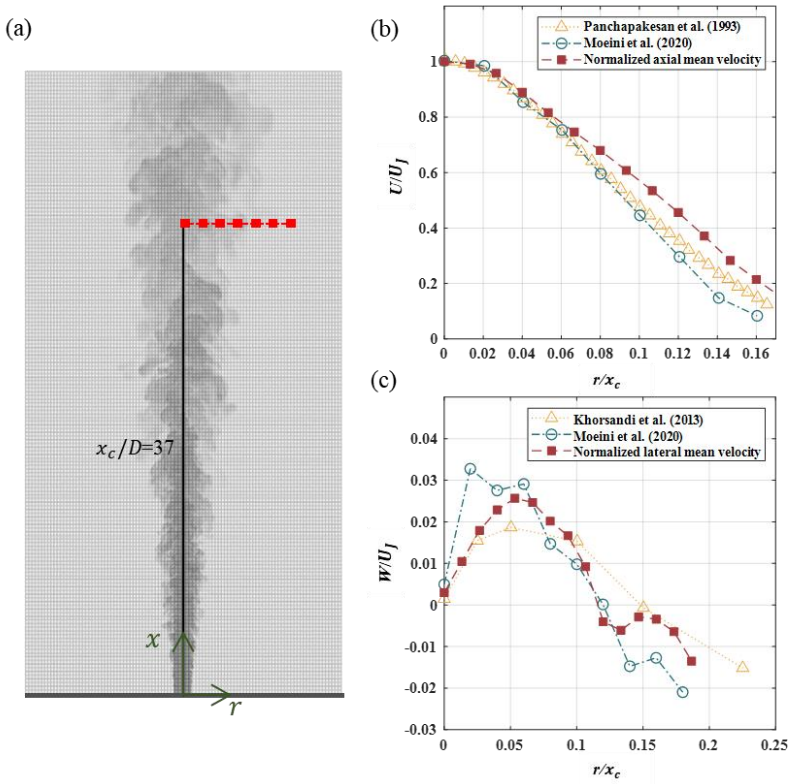

Figure 43. (a) Schematic view of jet flow, (b) normalized axial, and (c) normalized lateral mean velocity were taken at x_c/D=37 and compared to other laboratory experimental studies

This study also compared it with another experimental study by Fang et al. (2020). In their

laboratory experiment, they used a wave flume that had a length of 40 m, a width of 1.2 m, and a water depth of 1.2 m. The jet nozzle had a diameter of 0.01 m and was positioned 1.2 m above the flume bottom. In Case VS1, the jet flow was discharged into a quiescent water environment to establish the baseline flow characteristics. The experimental parameters, including the jet Reynolds number, Froude number, and Weber number, were measured and utilized to validate the proposed integral model for a non-buoyant turbulent jet in a wave environment. The second case in this was validated with Case VS1, which was a non-buoyant jet experiment. Figure 44 shows the axial velocity of the jet normalized by the jet inlet velocity along the centerline axis. Although the numerical results are slightly different from the laboratory experiments, the R-squared value is approximately 0.928. This indicates that the numerical experiments are in good agreement with the laboratory experiments conducted by Fang et al. (2020).

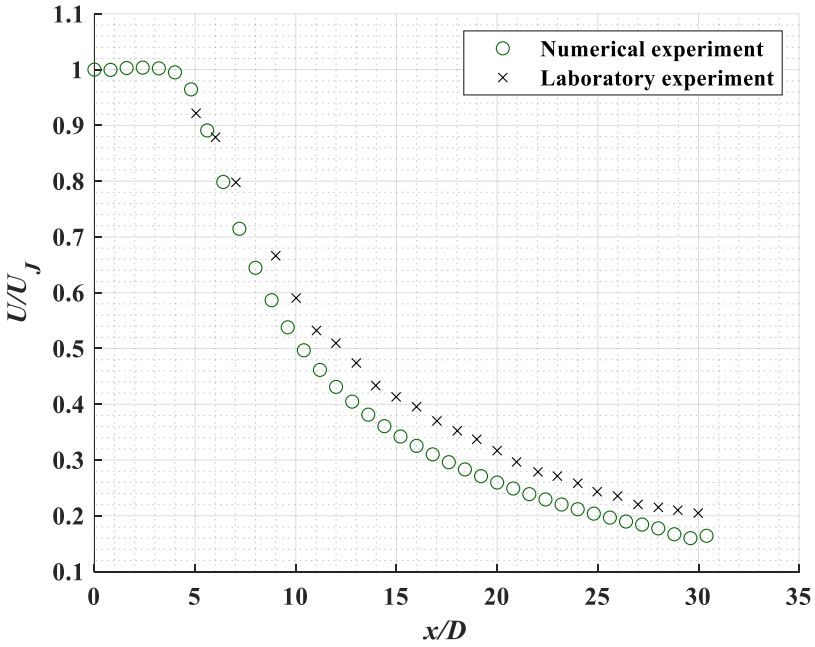

Figure 44. The axial velocity of the jet normalized by the jet inlet velocity along the centerline axis of the jet.

These well-matched results with laboratory experiments allowed for an analysis of the jet flow using

the same methods as in the previous jet flow study. Figure 45 shows the average axial velocity profile at two points along the jet, illustrating how the jet decays and spreads as the axial distance increases. The jet decay is clearly visible as a decrease in the centerline velocity, $< U >$. The axial velocity profiles also indicate that the velocity is maintained at a wider range as it moves away from the jet flow. This implies that the value of $r_{1/2}$, which is half of the centerline mean flow velocity, moves away from the centerline. Additionally, in Figure 46, where the profile is plotted for $r/r_{1/2}$ and reduced to a single curve, the trend of the average shaft velocity profile remains unchanged, confirming that the self-similarity of the jet flow is well-preserved. As previously mentioned, self-similarity is achieved in the ZEF when $x/D > 6.2$. In this study, this property is observed at a distance approximately 0.12 m away from the jet inlet.

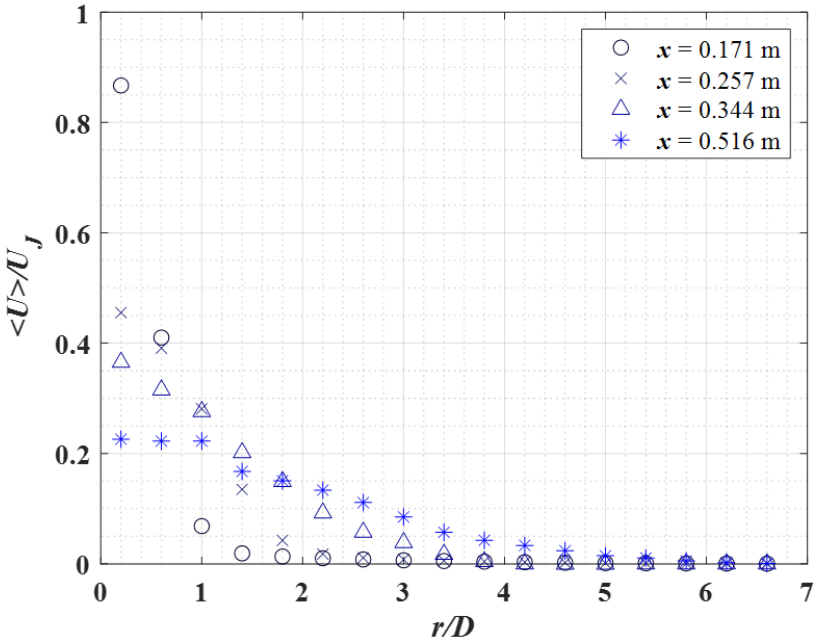

Figure 45. Radial profiles of mean axial velocity in a turbulent round jet at Reynolds number of 10,000.

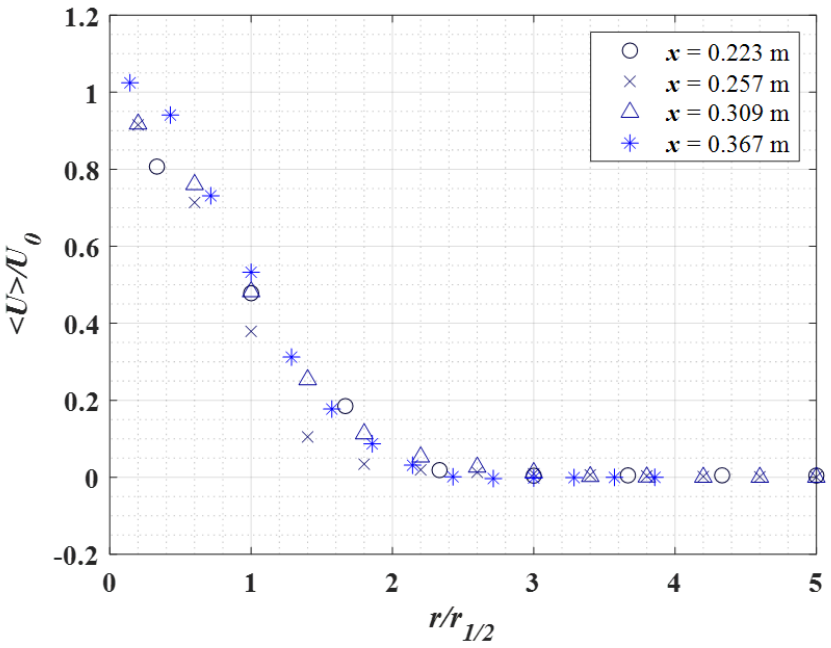

Figure 46. Radial profiles of mean axial velocity against radial distance normalized $r_{1/2}$ in a turbulent round jet, Reynolds number of 10,000.

Figure 47 illustrates the reciprocal of U_0 plotted against x/D. The experimental data is observed to fall on a straight line that can be represented by the following equation: $\frac{U_0(x)}{U_J} = \frac{B}{(x-x_0)/D}$. The empirical constant B can be calculated using this equation, and summarized in Chapter 2. Table 16 presents a comparison with other experiments. The spreading rate, also known as S, is obtained from the equation, $r_{1/2}(x) = S(x - x_0)$, and is summarized in Table 16.

Table 16. The spreading rate S and velocity decay constant B for turbulent round jets (Pope, 2000) and present work.

	Panchapakesan and Lumley (1993)	Hussein et al. (1994), hot-wire data	Hussein et al. (1994), laser-Doppler data	Present work
Re	11,000	95,500	95,500	10,000
S	0.096	0.102	0.094	0.116
B	6.06	5.9	5.8	5.01

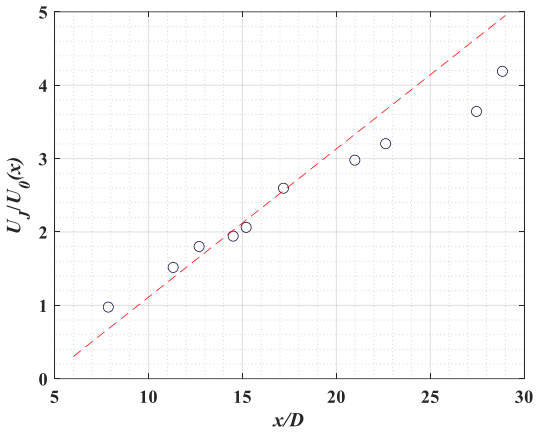

Figure 47. The inverse of the mean centerline velocity along the centerline in a turbulent round jet at Reynolds number of 10,000

Figure 48 displays the temporal velocity spectra of the axial and lateral velocity, constructed based on the fluctuating velocities obtained from LES for Case 1 at x_c/D=22.5. The spectra were obtained at a frequency of 100 Hz, with the maximum value of the energy spectrum limited to 50 Hz due to the Nyquist criterion. The axial and lateral velocity spectra clearly show the energy-containing area and energy decay parts. It has been observed that both energy ranges fall between $10^{-14}m^2/s^2$ and $10^{-4}\ m^2/s^2$. This indicates that the axial and lateral directions contain the same amount of energy and decay at the same rate.

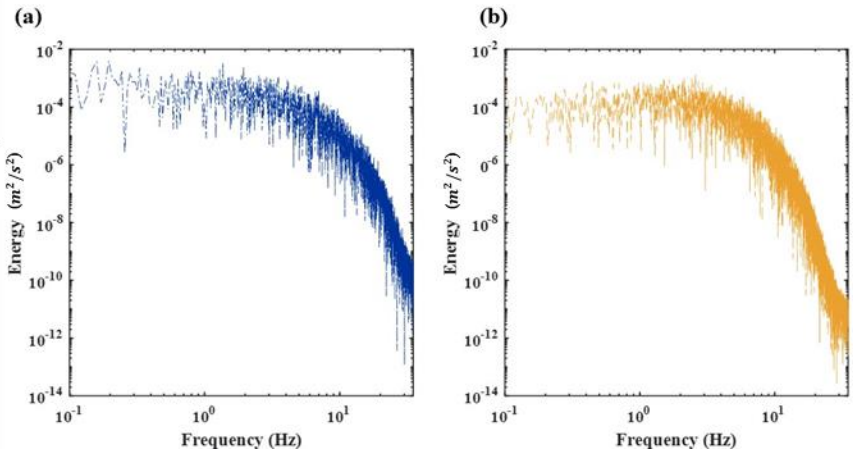

Figure 48. (a) Axial and (b) lateral velocity spectra at the jet centerline for the 100 Hz sampling frequencies measured at y_c/D=22.5

In this study, the data was calculated using LES (Large Eddy Simulation) for three cases, namely Case 1, Case 2, and Case 3, with grid sizes of 0.5, 0.6, and 0.7 cm, respectively. The values used in these cases were referred to as High Resolution (HR), Intermediate Resolution (IR), and Low Resolution (LR), respectively. Figure 49 depicts the energy spectra of HR, IR, and LR LES data, represented in green, yellow, and purple, respectively. The spatial energy spectra were calculated using velocity fluctuations and were derived at each time, followed by time-averaging the spectra obtained over a total of 300 seconds. The energy range is approximately from 10^-15 to 10^-6 m^2/s^2, and the same range can be observed in all three graphs. It can be observed that as the grid size decreases, the maximum wave number also decreases.

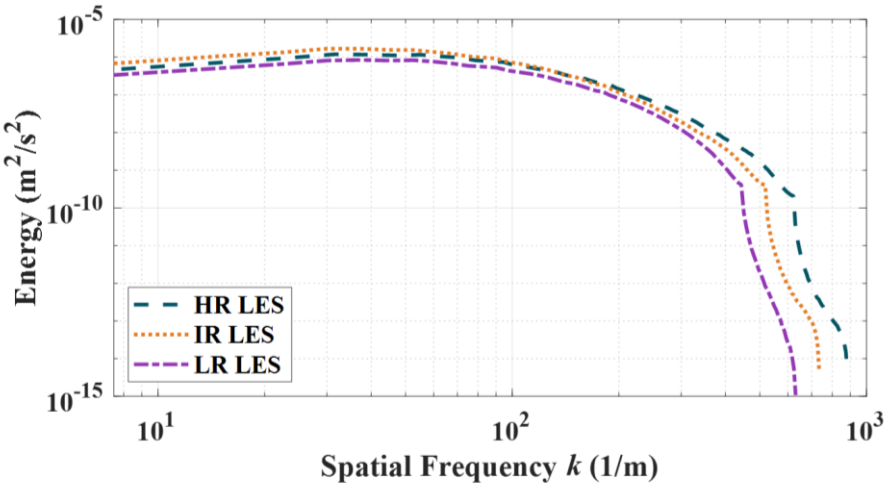

Figure 49. Spatial energy spectrum of high, intermediate, and low resolution LES

5.2 Convolutional Neural Network

The CNN algorithm extracts image features and generates data while preserving the characteristics of the images, as mentioned in Section 4.3. There are two CNN architectures described. The first proposed CNN architecture, known as the most basic algorithm, consists of a single-path layer with

convolutional layers, nonlinear functions, and padding. The second CNN architecture is composed of three paths. In this study, there are four CNN results, as shown in Table 17. Comparing one path CNN + IR LES with three path CNN + IR LES allows for the observation of differences in results based on the CNN algorithm. The results obtained by predicting IR and LR LES using the three path CNN demonstrate the ability to accurately learn SGS flow behavior even as the grid size of LES increases. This analysis was conducted to verify that the SGS flow behavior can be accurately captured regardless of the increasing grid size of LES. Finally, the data trained with intermediate and low Reynolds numbers is utilized to predict high Reynolds numbers, verifying whether the CNN model generates accurate results across various Reynolds numbers.

Table 17. Classification of CNN results

Classification	CNN architecture	Train case	Test case
One path CNN + IR LES	One path CNN	Case 1	Case 2
Three path CNN + IR LES	Three path CNN	Case 1	Case 2
Three path CNN + LR LES	Three path CNN	Case 1	Case 3
Three path CNN + Low and Intermediate Reynolds number	Three path CNN	Case 3 and 4	Case 5

The energy spectrum presented in Figure 50 illustrates the data reconstructed using the one path CNN technique for IR LES data and the data reconstructed using the three path CNN technique for IR LES data, both aimed at reproducing the velocity field of HR LES. In addition, to facilitate data comparison, the energy spectrum of the actual HR LES data has also been plotted. All energy spectrums were derived in a manner similar to Figure 49. The spectrum is depicted in blue, magenta, and green colors, respectively. As observed in the previously examined energy spectrum in Figure 49, the maximum wave number of the IR LES was smaller than that of the HR LES. However, by utilizing CNN techniques, it can be confirmed that the maximum wave number of the IR LES aligns with the

maximum wave number of the HR LES. When using the one path CNN, it was observed that the energy values were lower compared to the HR LES energy. On the other hand, when using the three path CNN, it was noted that the energy spectrum had a similar slope to the HR LES energy spectrum. It was observed that the three path CNN more accurately reproduces the spectrum compared to the one path CNN. Subsequently, the time averaged velocity and turbulence kinetic energy (TKE) fields were depicted based on the results obtained from the three path CNN only.

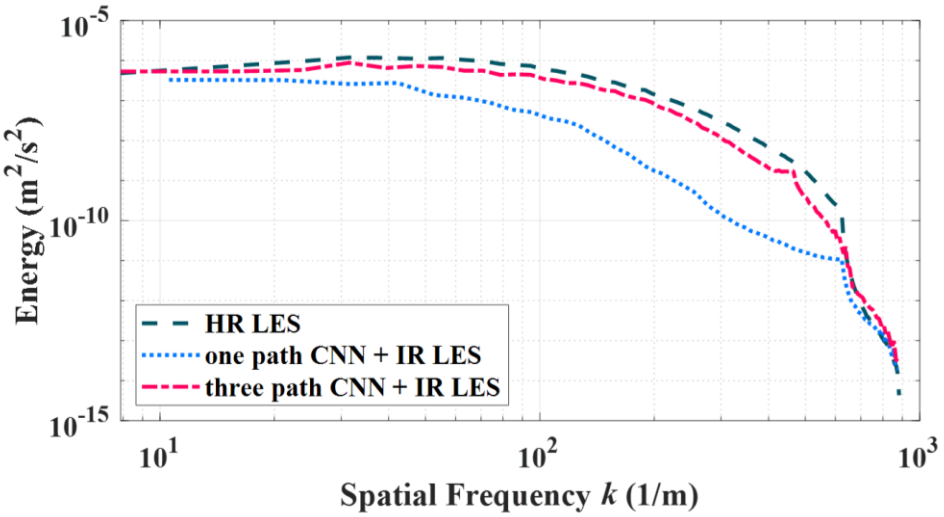

Figure 50. Spatial energy spectrum of high resolution LES, one path CNN + intermediate resolution LES, and three path CNN + intermediate resolution LES

Figure 51 depicts the results of mean velocity fields obtained by applying the three path CNN with varying grid sizes. The figure consists of three subfigures: (a) shows the mean velocity field of HR LES, (b) represents the result generated using the three path CNN technique for IR LES data, and (c) represents the result generated using the three path CNN technique for LR LES data. The mean data was obtained by averaging data collected over a period of 5 minutes. The velocity is visualized using a color gradient ranging from blue to yellow, with yellow indicating higher velocities. Notably, high

velocities are observed near the jet inlet. In Figure (b), which represents the velocity field generated using the three path CNN and IR LES data, the corresponding values of R^2 and NRMSE were calculated as 0.9811 and 1.94%, respectively. Similarly, in Figure (c), which represents the velocity field generated using the three path CNN and LR LES data, the corresponding values of R^2 and NRMSE were calculated as 0.9628 and 2.66%, respectively.

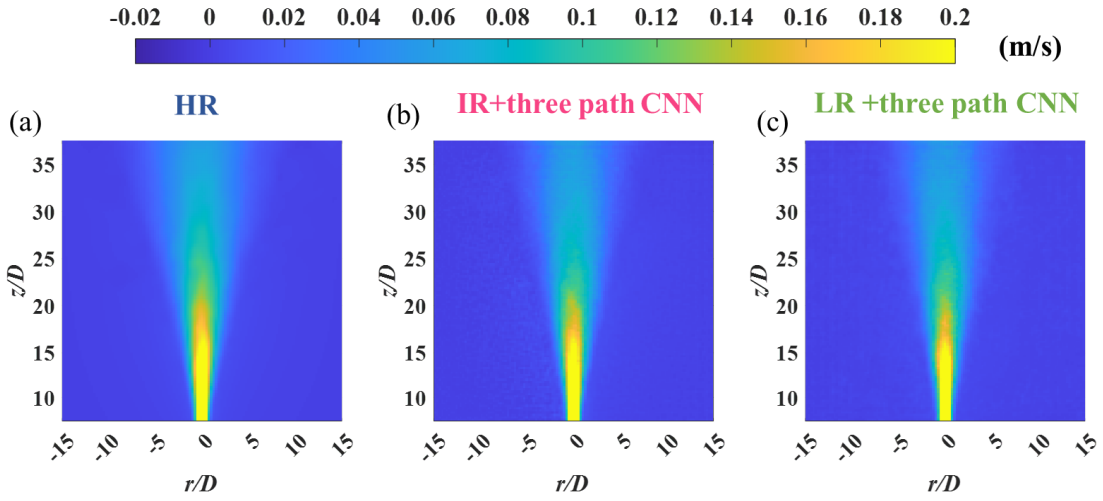

Figure 51. The time-averaged magnitude of velocity; (a) High resolution LES, (b) Intermediate resolution LES + three path CNN, and (c) Low resolution LES + three path CNN

Subsequently, TKE was calculated and depicted as shown in Figure 51. The TKE values are obtained from HR LES, IR LES with the three path CNN, and LR LES with the three path CNN, arranged from left to right in Figure 51. Similar to the previous figures, the color gradient in the figure represents increasing TKE values from blue to yellow, with stronger TKE values observed near the inlet compared to the surrounding regions. For the case of IR LES with a 10% increase in grid size and the utilization of the three path CNN to generate the velocity field, the TKE was calculated as 0.9260, with an RMSE of 11.04%. In Figure 51c, where the grid size increased by 20% and the three path CNN was

used to generate the velocity field, the TKE was calculated as 0.9033, with an RMSE of 12.41%.

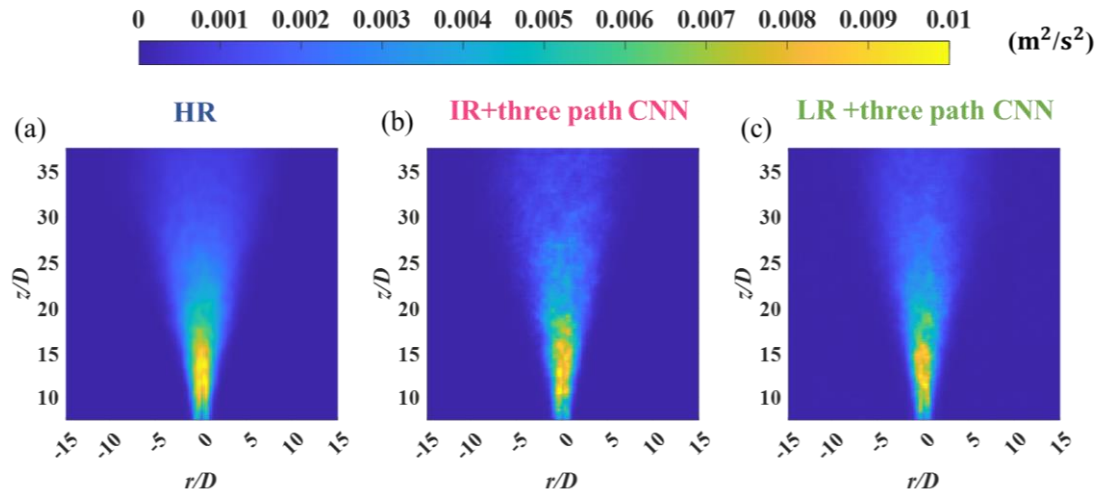

Figure 52. TKE of LES and CNN results; (a) High resolution LES, (b) Intermediate resolution LES + three path CNN, and (c) Low resolution LES + three path CNN

Figure 53a and b depict the time averaged velocity and TKE, respectively, calculated using the azimuthal averaging technique. The HR LES data is represented by circles, IR LES data with the three path CNN by squares, and LR LES data with the three path CNN by triangles. Both velocity and TKE values are nondimensionalized using the central values, and the distance from the center is nondimensionalized using the diameter of the jet. In Figure 53a, the strongest velocity occurs at the centerline, and the velocity decreases as the distance from the centerline increases. Approximately at a distance of 7.5 cm from the centerline, the velocity reaches zero. In Figure 53b, the TKE exhibits the highest value at the centerline, and the TKE decreases as the distance from the centerline increases. TKE value reaches zero at a distance of 9.5 cm from the centerline. Both graphs show good agreement between HR LES, IR LES with the three path CNN, and LR LES with the three path CNN.

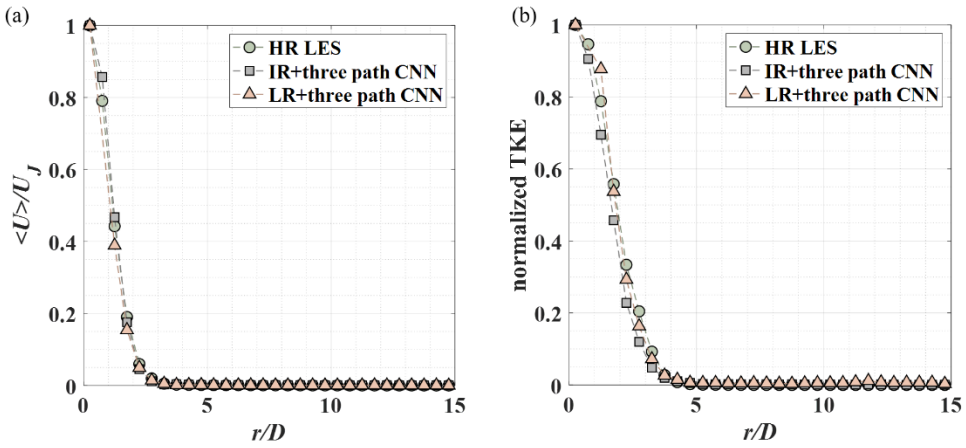

Figure 53. (a) Azimuthal temporally averaged velocity profile normalized by jet inlet velocity and (b) Azimuthal averaged turbulence kinetic energy normalized by jet inlet velocity.

The R-squared values and NRMSE for the temporally averaged velocity profiles and TKE in the entire region have been summarized in Table 18. It includes the results obtained using the one path CNN on IR LES, the results obtained using the three path CNN on IR LES, and the results obtained using the three path CNN on LR LES. It can be observed that all the results obtained using the three path CNN have an R-squared value exceeding 0.9. On the other hand, the results obtained using the one path CNN exhibit an R-squared value of 0.575 for the velocity and negative values for the TKE. This indicates that the results obtained using the three path CNN architecture more accurately capture the desired characteristics and can be considered as better method in this context.

Table 18. R-squared and NRMSE of CNN results compared to the ground truth

		One path CNN + IR LES	Three path CNN + IR LES	Three path CNN + LR LES
R-squared	Temporally averaging	0.575	0.981	0.963
NRMSE		7.88 %	1.94 %	2.66 %
R-squared	TKE	-0.6886	0.926	0.903
NRMSE		50.85 %	11.04 %	12.41 %

Figure 54 depicts the time series of the instantaneous velocity along the r_1, x, and r_2 axes for both the LES and the three path CNN on IR LES, obtained at the center of the computational domain. Both LES and three path CNN on IR LES show time-averaged instantaneous velocities of 0 m/s along the r_1 axis, 0.05 m/s along the x axis, and 0 m/s along the r_2 axis. As expected, based on the TKE, the velocity range in three path CNN on IR LES is smaller compared to LES. Specifically, in three path CNN on IR LES, the maximum and minimum values along the x axis are -0.06 m/s and 0.06 m/s, respectively, whereas in LES, the corresponding values are 0.08 m/s and 0.08 m/s. This difference can be attributed to the fact that although the trained model accurately reproduces the velocity field, the predicted values for TKE are slightly lower.

In order to provide a more detailed comparison between the results obtained using the one path CNN on IR LES, the three path CNN on IR LES, and the three path CNN on LR LES, a Taylor diagram was utilized for evaluation (Figure 55). The Taylor diagram visualizes the standard deviation, correlation coefficient, and root mean square error between the reference values and the model results, enabling a comprehensive assessment of model performance. On the left side, the lower values of the standard deviation indicate a better match between the model results and the reference data. Along the radial lines on the right side, a stronger correlation with the reference data is indicated by values closer to the bottom. The concentric circles inside the diagram represent the root mean square error, and results closer to the ground truth indicate better performance. From the diagram, it can be observed that the one path CNN on IR LES is positioned in the top-left corner, indicating a lower correlation coefficient of 0.2. On the other hand, the three path CNN on IR LES and the three path CNN on LR LES show correlation coefficients of 0.9 or higher, along with root mean square errors of approximately 0.0005. The standard deviations are also calculated to be around 0.0005. Overall, the Taylor diagram confirms that the results obtained using the three path CNN on IR LES and LR LES exhibit higher correlation and lower errors compared to the one path CNN on IR LES, demonstrating better agreement with the ground truth data.

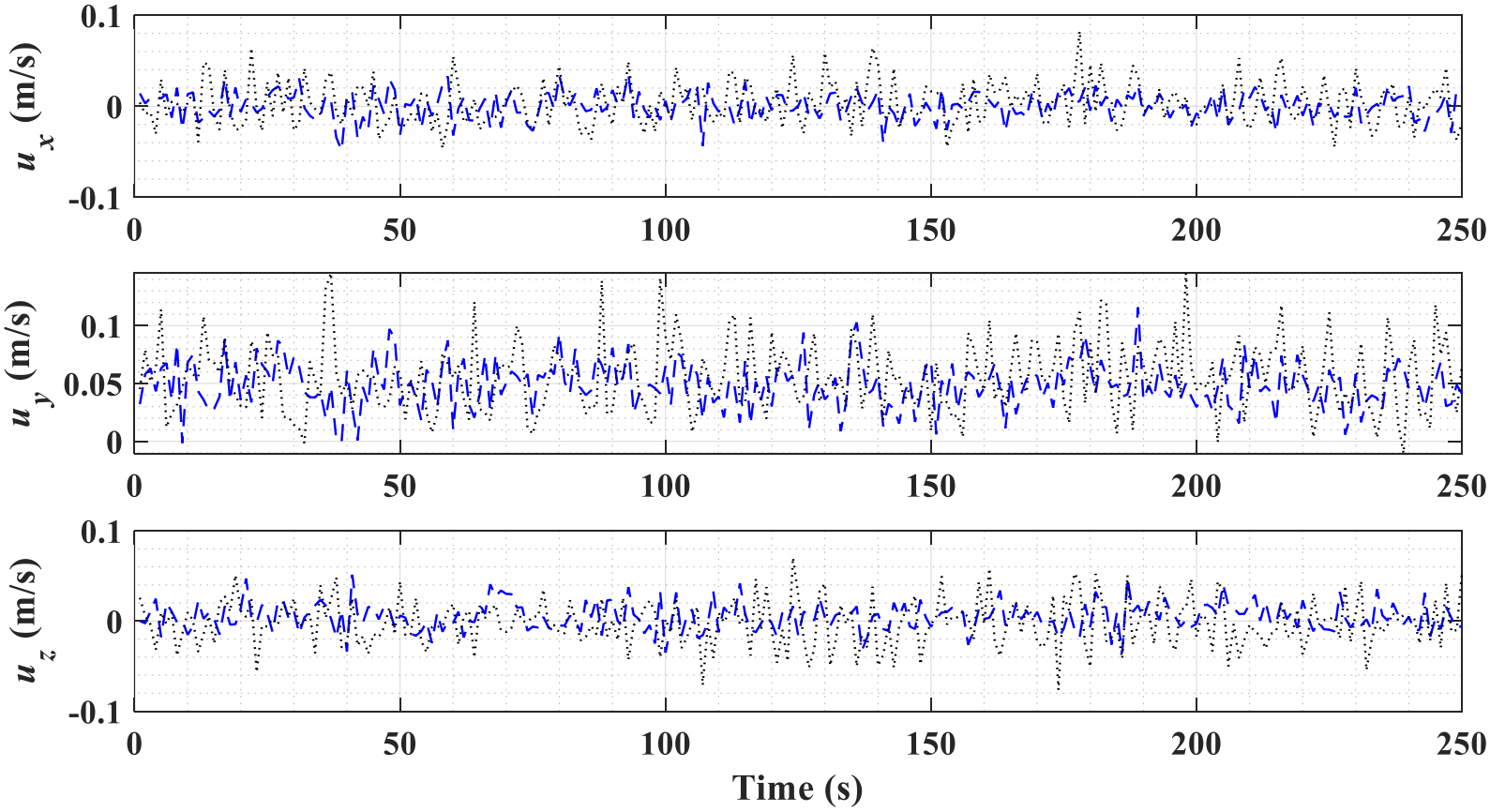

Figure 54. Instantaneous velocities of LES (black line) and the three path CNN result on IR LES (blue line) at center of domain

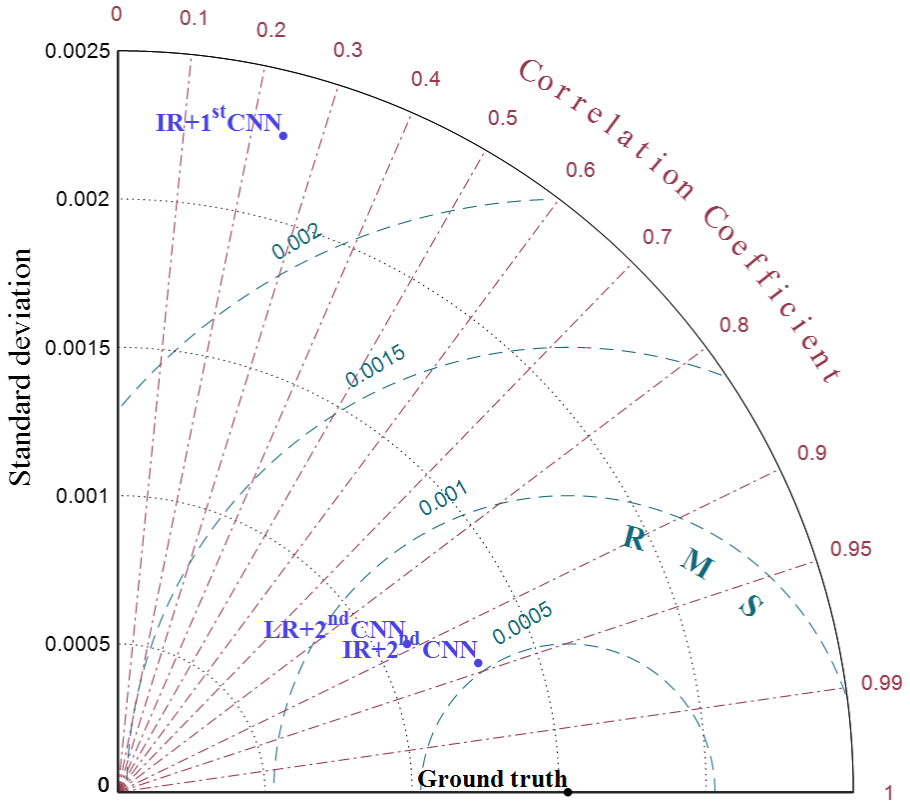

Figure 55. Taylor diagram displaying a statistical comparison with the results obtained using the one path CNN on IR LES, the three path CNN on IR LES, and the three path CNN on LR LES and the ground truth data.

The following is an investigation into whether the three path CNN can accurately generate velocity fields despite variations in Reynolds number. Figure 56a represents the numerical results obtained from LES at a high Reynolds number of 20,000. Figure 56b illustrates the prediction of the velocity field at high Reynolds number using the three path CNN trained on low and intermediate Reynolds numbers. The mean velocity field was obtained by averaging the results over a 5-minute period. As observed in the previous grid convergence study, the velocity is stronger near the jet inlet and decreases as the distance from the inlet increases. The R-squared value of 0.9003 and the RMSE of 4.03% indicate a strong agreement between the predicted velocity field generated by the three path CNN and the high

Reynolds number LES data.

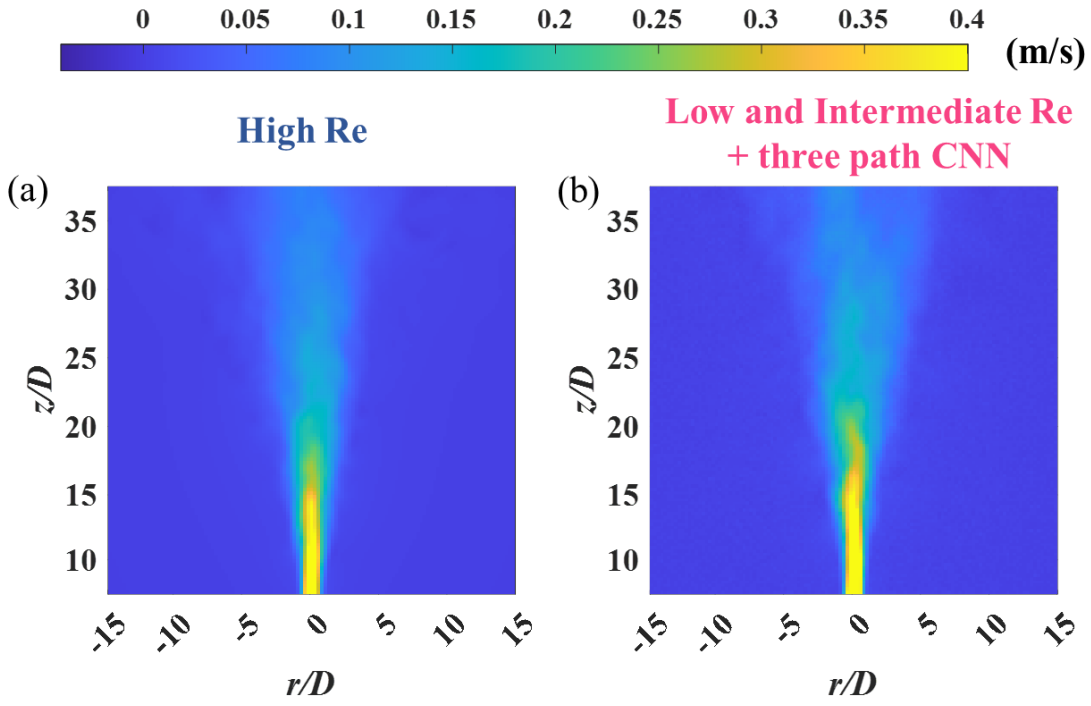

Figure 56. The time-averaged magnitude of velocity; (a) High Reynolds number and (b) Low and Intermediate Reynolds number + three path CNN

Figure 57a represents the TKE obtained from LES at a high Reynolds number of 20000. Figure 57b depicts the TKE obtained by utilizing the model trained using low and intermediate Reynolds numbers with the three path CNN to predict the velocity field at the high Reynolds number. The R-squared value is 0.673, and the RMSE is 1.61%. In this study, the three path CNN model is considered to be superior in generating flow velocity compared to the one path CNN. This suggests that the three path CNN is a more effective technique in capturing spatial features.

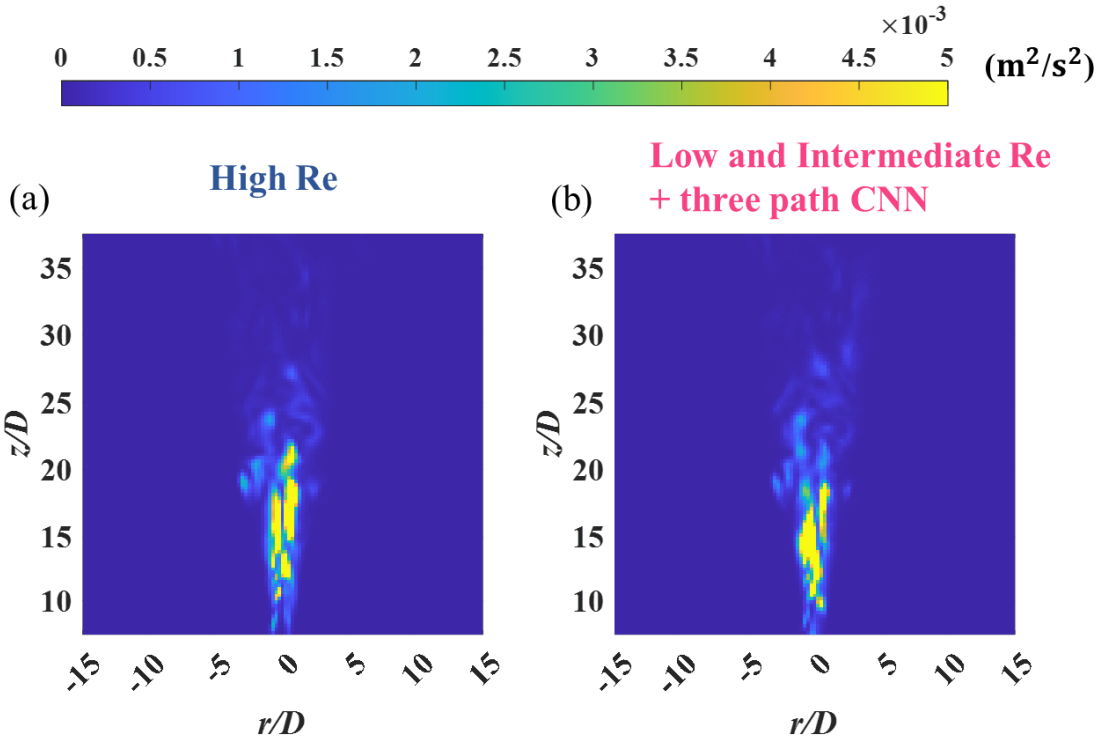

Figure 57. TKE (a) High Reynolds number and (b) Low and Intermediate Reynolds number + three path CNN

5.3 Generative Adversarial Network

GAN is a type of unsupervised learning that can learn without labels and has a significant advantage in directly generating data. There are two GAN architectures used in generating velocity fields, as described in section 4.3. The first architecture, named spatially isolated GAN, is the most fundamental algorithm, which connects all data using a fully connected layer for training all data at once. The second architecture, named spatially connected GAN, uses convolutional layers to reconstruct high-resolution flow fields from low-resolution ones. In this study, four GAN outcomes are presented, as summarized in Table 19. By comparing the outcomes of the spatially isolated GAN + IR LES configuration with those of the spatially connected GAN + IR LES configuration, it can be observed the differences in results based on the GAN algorithm. The results obtained through the utilization of the spatially connected GAN on IR and LR LES demonstrate the ability to accurately learn flow behavior even as the grid size of LES increases. Moreover, the trained data encompassing intermediate and low Reynolds numbers are employed to generate the velocity field at high Reynolds numbers, thereby verifying the GAN model's ability to generate accurate results across a wide range of Reynolds numbers.

Table 19. Classification of GAN results

Classification	GAN architecture	Train case	Test case
Spatially isolated GAN + IR LES	Spatially isolated GAN	Case 1	Case 2
Spatially connected GAN + IR LES	Spatially connected GAN	Case 1	Case 2
Spatially connected GAN + LR LES	Spatially connected GAN	Case 1	Case 3
Spatially connected GAN + Low and Intermediate Reynolds number	Spatially connected GAN	Case 3 & 4	Case 5

The energy spectrum depicted in Figure 58 represents the outcomes obtained from HR LES data, spatially isolated GAN applied to IR LES data, and spatially connected GAN applied to IR LES data,

indicated by the colors green, blue, and red respectively. As shown in Figure 49, it is evident that the maximum wavenumber of IR LES is smaller than that of HR LES. However, with the implementation of the GAN technique, specifically in Figure 58, the maximum wavenumber of IR LES increases and aligns with that of HR LES. By employing the spatially isolated GAN technique, it becomes apparent that an energy cascade does not occur, resulting in a linear energy spectrum formation. Conversely, with the implementation of the spatially connected GAN, an energy cascade is observed. However, a notable increase in the energy spectrum can be observed compared to HR LES when the wavenumber ranges from 300 to 653. In this study, the velocity field and TKE were derived using the spatially connected GAN, which yields relatively consistent energy spectra.

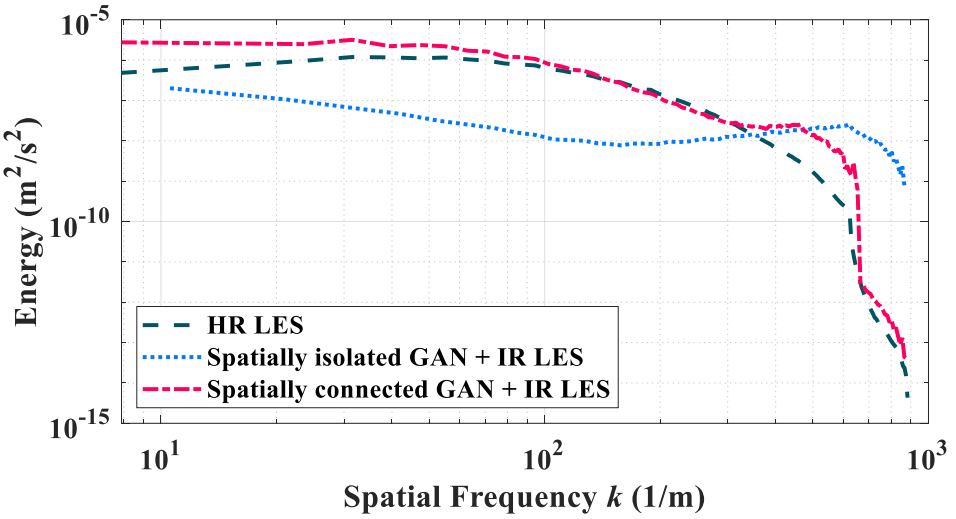

Figure 58. Spatial energy spectrum of high resolution LES, spatially isolated GAN + intermediate resolution LES, and spatially connected GAN+ intermediate resolution LES

The results based on changes in grid size are presented in Figure 59. The figure illustrates the mean velocity fields obtained from HR LES, the mean velocity field computed using the spatially connected GAN technique on IR LES data, and the mean velocity field obtained by applying the spatially

connected GAN technique to LR LES data. It can be observed that a high-velocity field is prominently present near the jet inlet. In Figure 59b, the mean velocity field generated using the spatially connected GAN technique on IR LES data yields R squared value of 0.9384 and NRMSE of 3.35%. Similarly, in Figure 59c, the mean velocity field obtained by applying the spatially connected GAN technique to LR LES data results in R squared value of 0.9270 and NRMSE of 3.54%.

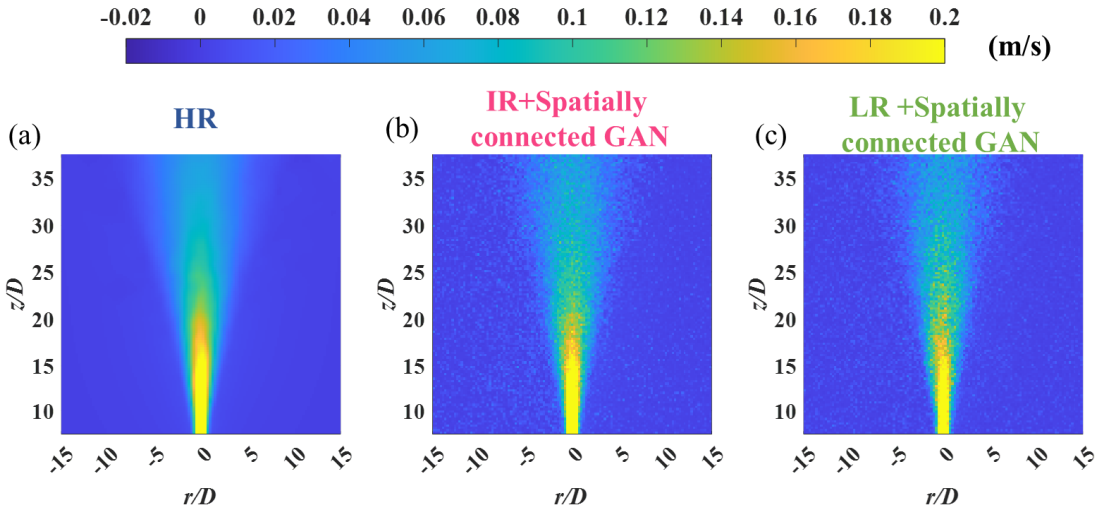

Figure 59. The time-averaged magnitude of velocity; (a) High resolution LES, (b) Intermediate resolution LES + spatially connected GAN, and (c) Low resolution LES + spatially connected GAN

The TKE is calculated and illustrated in Figure 60, utilizing HR LES, HR data obtained by applying the spatially connected GAN technique on IR LES, and HR data obtained by applying the spatially connected GAN technique on LR LES. The TKE is calculated and presented in Figure 60, where the grid size of the IR LES is increased by 20% compared to the HR LES grid size. The spatially connected GAN is then employed on the IR LES to generate the velocity field. The TKE is calculated as R-squared of 0.1939 with NRMSE of 36.08%. In Figure 60c, where the grid size is increased by 40% of the HR LES grid size to generate the velocity field, the calculated TKE is 0.0742 with an RMSE of 39.45%.

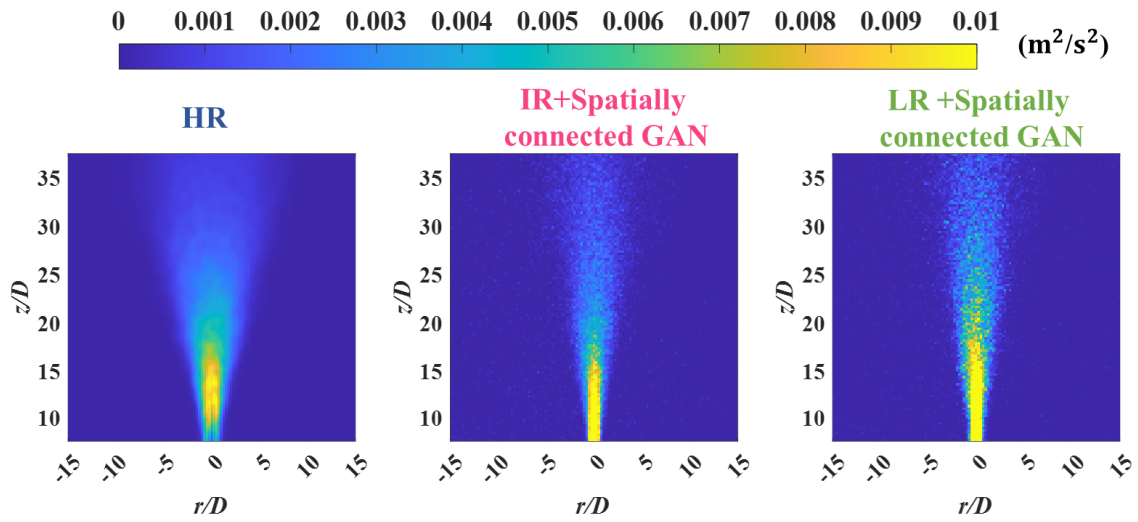

Figure 60. TKE of LES and GAN results; (a) High resolution LES, (b) Intermediate resolution LES + spatially connected GAN, and (c) Low resolution LES + spatially connected GAN

Figure 61a and b present the time-averaged velocity and TKE, respectively, calculated using the azimuthal averaging technique. The HR LES data is represented by circles, the IR LES data obtained with the spatially connected GAN is represented by squares, and the LR LES data obtained with the spatially connected GAN is represented by triangles. Both velocity and TKE values are normalized using the central values, and the distance from the center is normalized using the diameter of the jet. In Figure 61a, the centerline exhibits the highest velocity, which gradually decreases as the distance from the centerline increases. The velocity reaches zero at approximately 7.5 cm from the centerline. In Figure 61b, the TKE reaches its maximum value at the centerline and decreases as the distance from the centerline increases. Notably, the TKE value reaches zero starting from a distance of 9.5 cm in the HR LES, while it converges to zero at a distance of 6.5 cm. These findings suggest that there is a slight discrepancy in the TKE compared to the ground truth.

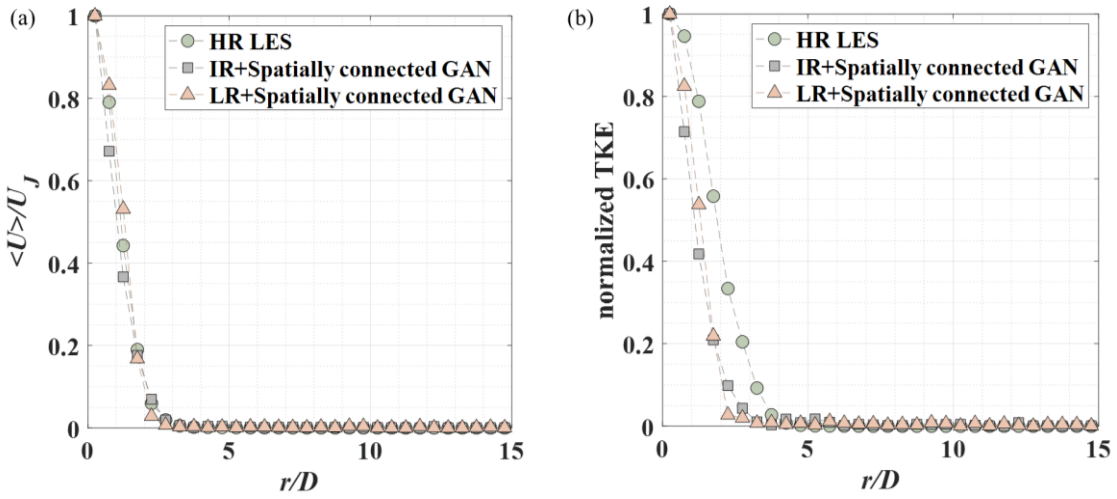

Figure 61. (a) Azimuthal temporally averaged velocity profile normalized by jet inlet velocity and (b) Azimuthal averaged turbulence kinetic energy normalized along the x-axis at x/D=6.7.

Table 20 summarizes the R-squared values and NRMSE for the temporally averaged velocity profiles and TKE across the entire region. This table includes results obtained using the spatially isolated GAN on IR LES, results obtained using the spatially connected GAN on LR LES, and results obtained using the spatially connected GAN on LR LES. It is observed that all the results obtained using the spatially connected GAN exhibit an R-squared value exceeding 0.9. However, the TKE values show relatively lower R-squared values compared to the velocity profiles.

Table 20. R-squared and normalized RMSE of GAN results with ground truth

		Isolated GAN + IR LES	Connected GAN + IR LES	Connected GAN + LR LES
R-squared	Temporally averaging	0.8223	0.9384	0.9270
NRMSE		0.0484	3.35 %	3.54 %
R-squared	TKE	-0.029	0.194	0.074
NRMSE		0.4152	36.08 %	39.45 %

Figure 62 illustrates the time series of the instantaneous velocity obtained at the center of the domain along the x and r axes for LES and HR data using the spatially connected GAN on IR LES. Both LES and HR data exhibit an approximate time-averaged instantaneous velocity of 0, 0.05, and 0 m/s along the x and r axes. Moreover, the velocity ranges of HR data and LES show similarity in each axis. The r-axes values reveal maximum and minimum velocities of -0.08 and 0.08 m/s for LES and HR data using the spatially connected GAN on IR LES, respectively. Along the x-axis, the velocity range of LES ranges from -0.02 to 0.18 m/s, which closely aligns with the range observed in HR data using the spatially connected GAN on IR LES.

The GAN technique was also analyzed using a Taylor diagram in Figure 63, as mentioned before, standard deviation, correlation coefficient, and root mean square error are plotted. The superior outcomes are depicted as we progress downward from the left to the right, approaching closer proximity to the ground truth at the bottom. According to the findings of this study, the correlation coefficient of the spatially connected GAN on IR LES is high, whereas the correlation coefficient of the spatially connected GAN on LR LES is relatively lower, at around 0.5. Additionally, when the spatially isolated GAN is used, both the correlation coefficient and standard deviation are lower. Based on the preceding findings, it has been determined that the spatially connected GAN is a more suitable approach for generating time-averaged velocity fields compared to the spatially isolated GAN. However, during this process, it was observed that the TKE values generated were smaller than the actual HR values. This indicates that the velocity fluctuations are underestimated compared to the ground truth.

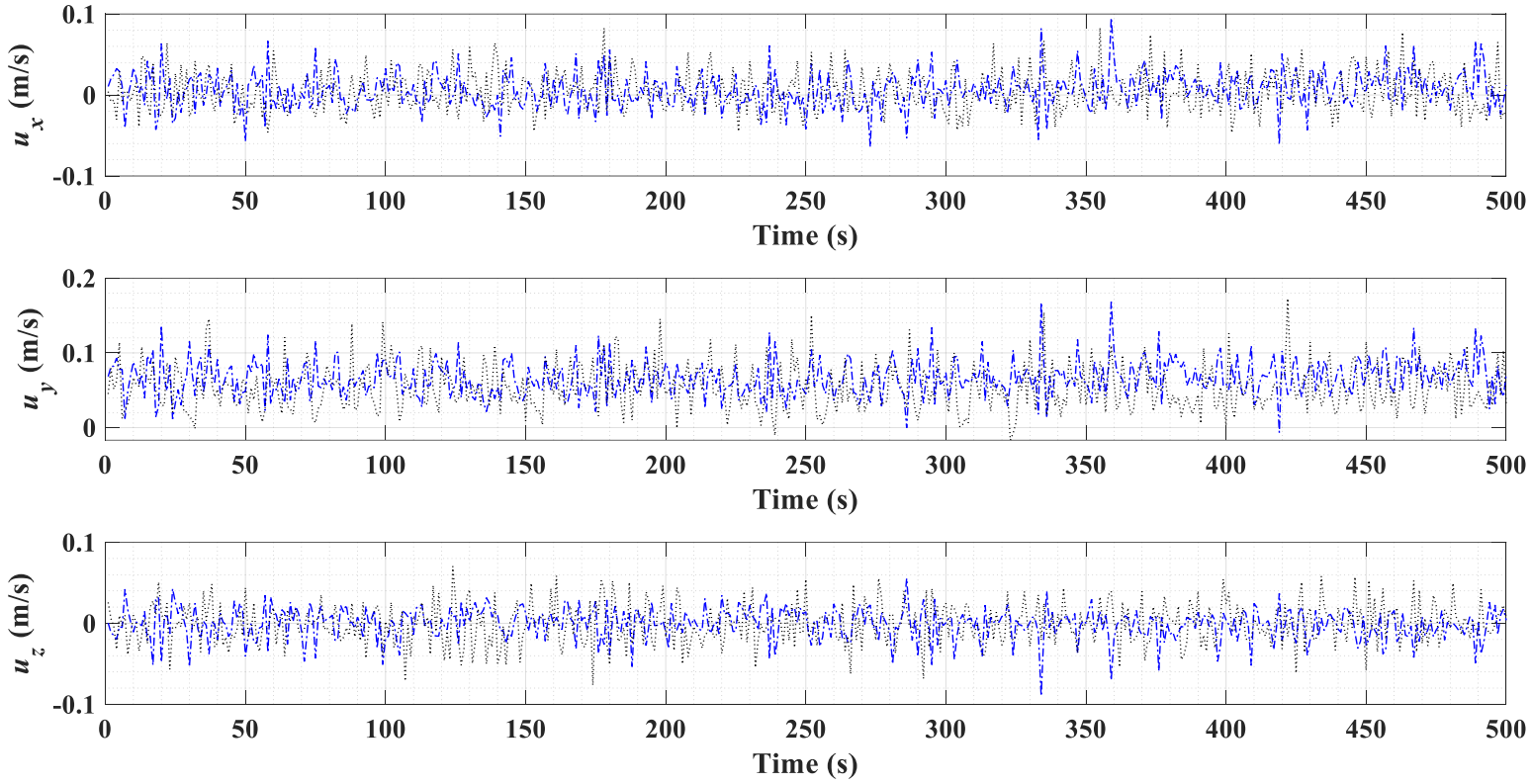

Figure 62. Instantaneous velocities of LES (black line) and the spatially connected GAN result on IR LES (blue line) at center of domain

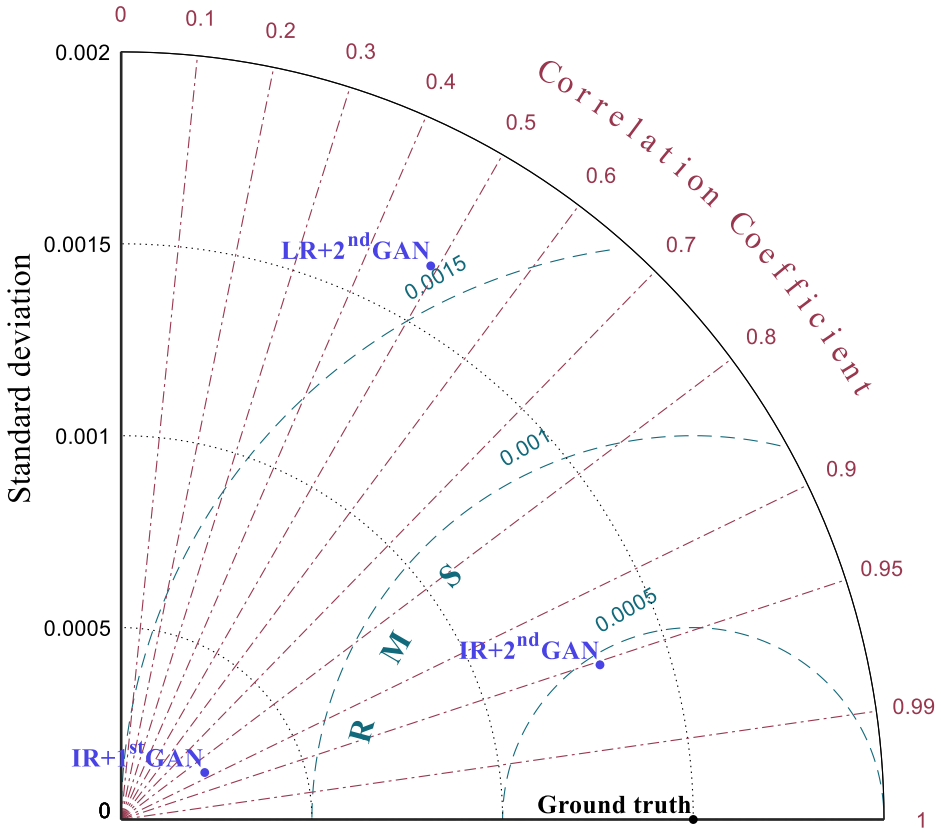

Figure 63. Taylor diagram of case HR LES data and GAN results

Figure 64a represents the numerical results obtained from LES at a high Reynolds number of 20000. On the right side, the spatially connected GAN model trained using low and intermediate Reynolds numbers is utilized to predict velocities at the high Reynolds number, resulting in the visualization of the mean velocity field. It can be observed that the velocity is high near the jet inlet and decreases as the distance from the inlet increases. The R squared value for this case is determined to be 0.8394, with NRMSE of 5.99%.

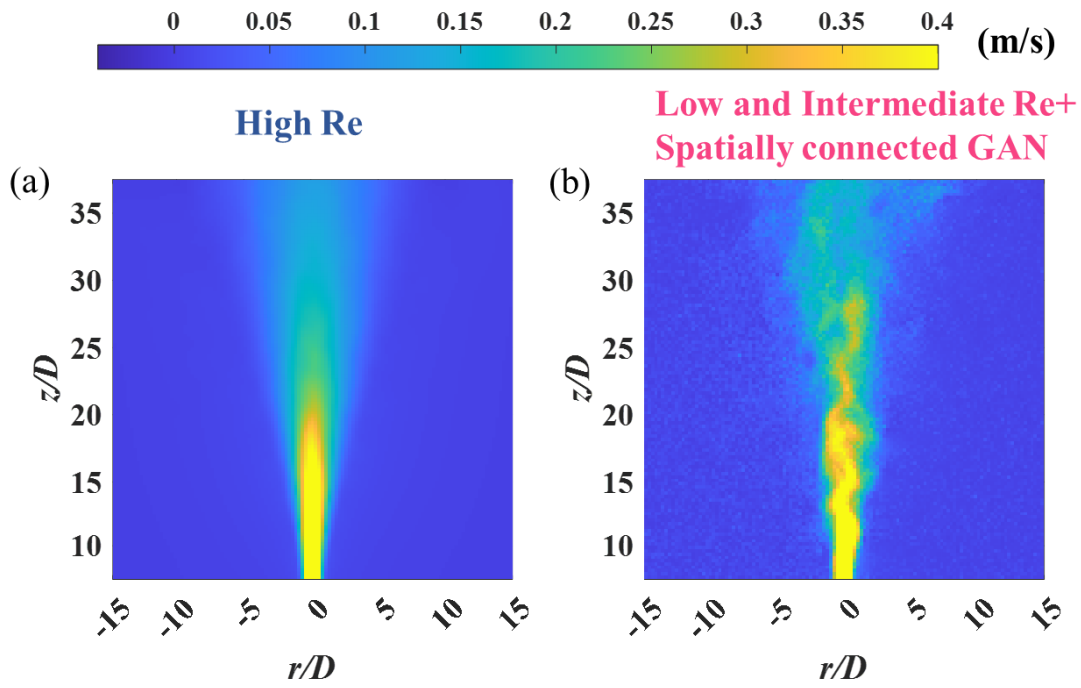

Figure 64. The time-averaged magnitude of velocity; (a) High Reynolds number, (b) Low and Intermediate Reynolds number + Spatially connected GAN

Figure 65a illustrates the numerical results obtained from LES at a high Reynolds number of 20000. In Figure 65b, the spatially connected GAN model trained using low and intermediate Reynolds numbers is employed to predict velocity fields at the high Reynolds number, resulting in the visualization of TKE. The R-squared value for this case is determined to be 0.3303, with NRMSE of 26.31%. This can be considered as one of the limitations of GAN known as mode collapse, which refers to the generator of GAN being biased towards a specific distribution or mode. Mode collapse often occurs when the training data exhibits similar patterns or characteristics. In this study, resolved velocity data excluding the subgrid scale velocity field are used as the input data. The exclusion of the random component of turbulence from the input data of GAN can lead to mode collapse, as the input data becomes relatively lacking variation.

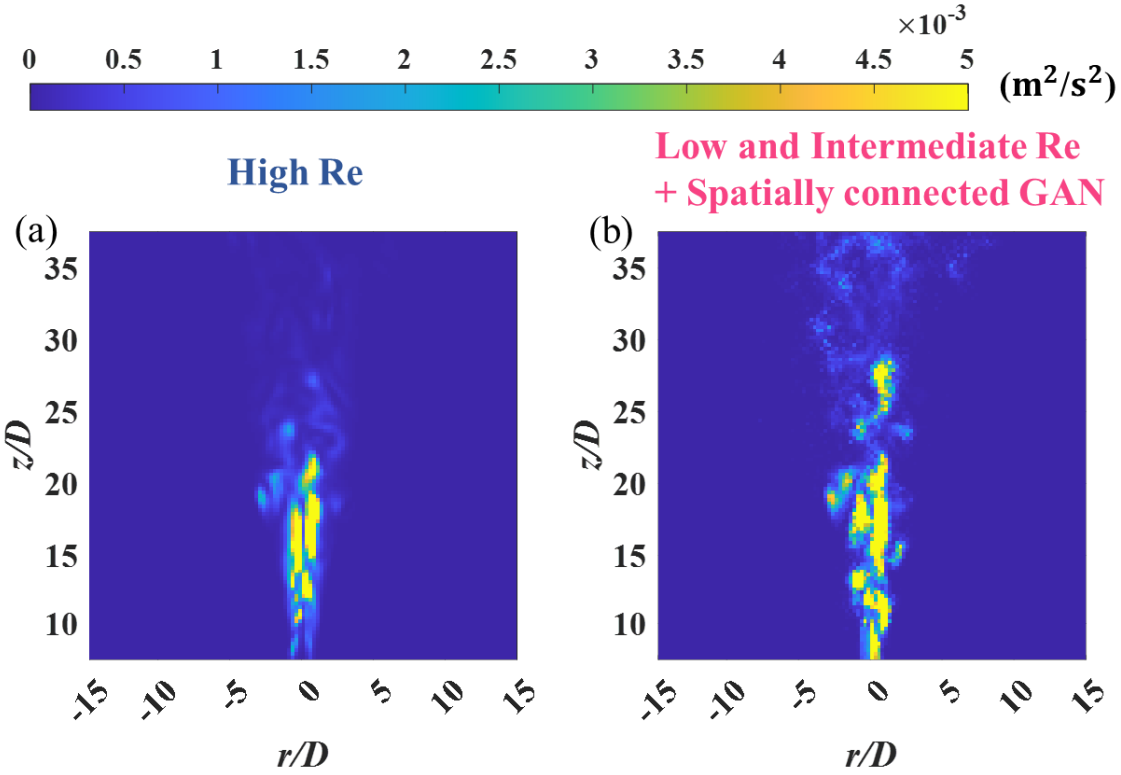

Figure 65. TKE (a) High Reynolds number and (b) Low and Intermediate Reynolds number + Spatially connected GAN

5.4 Time Averaged Turbulent Characteristic

The input data in this study consists of time-averaged velocities, as depicted in Figure 66. The mean streamwise velocity (U) is observed to be relatively higher compared to the radial velocities (V, W). As the distance from the jet inlet decreases (x approaches 0), the mean streamwise velocity increases, while the mean radial velocities exhibit positive and negative values in the free shear stress region. The input data in this study includes the time-averaged velocity values along with their corresponding grid point locations. The objective of this study is to predict time-averaged TKE, production, and dissipation using the provided input data. The output data is presented in Figure 67. It can be observed that the highest production and dissipation values are located near the starting point of the shear stress region, at approximately x/D is 0.2. The maximum value of TKE is measured to be 0.028 m^2/s^2. Generally, the production values outweigh the dissipation values, which is expected as this region is closer to the jet flow and experiences higher production rates. Moreover, TKE, production, and dissipation exhibit a distinct pattern near the jet flow and become more random as the distance from the jet inlet increases. The study aims to investigate the accuracy of ML techniques in simulating these turbulence characteristics.

The ML model was initially trained using a Reynolds number of 10,000, and subsequently tested using a Reynolds number of 15,000. The test input data, as shown in Figure 68, exhibits velocity field similar to that observed at a Reynolds number of 10,000, but with an overall increase in magnitude by a factor of 1.5. The shapes of TKE, production, and dissipation in the test output data resemble those observed at a Reynolds number of 10,000(Figure 69). The highest values of these variables are still located near the starting point of the shear stress region. The highest TKE value in the test output data is 0.0565 m^2/s^2. Consistent with the training data, the production values are larger than the dissipation values in the test output data, showing a stronger production near the jet inlet. All the time-averaged values are based on 30 second simulation time.

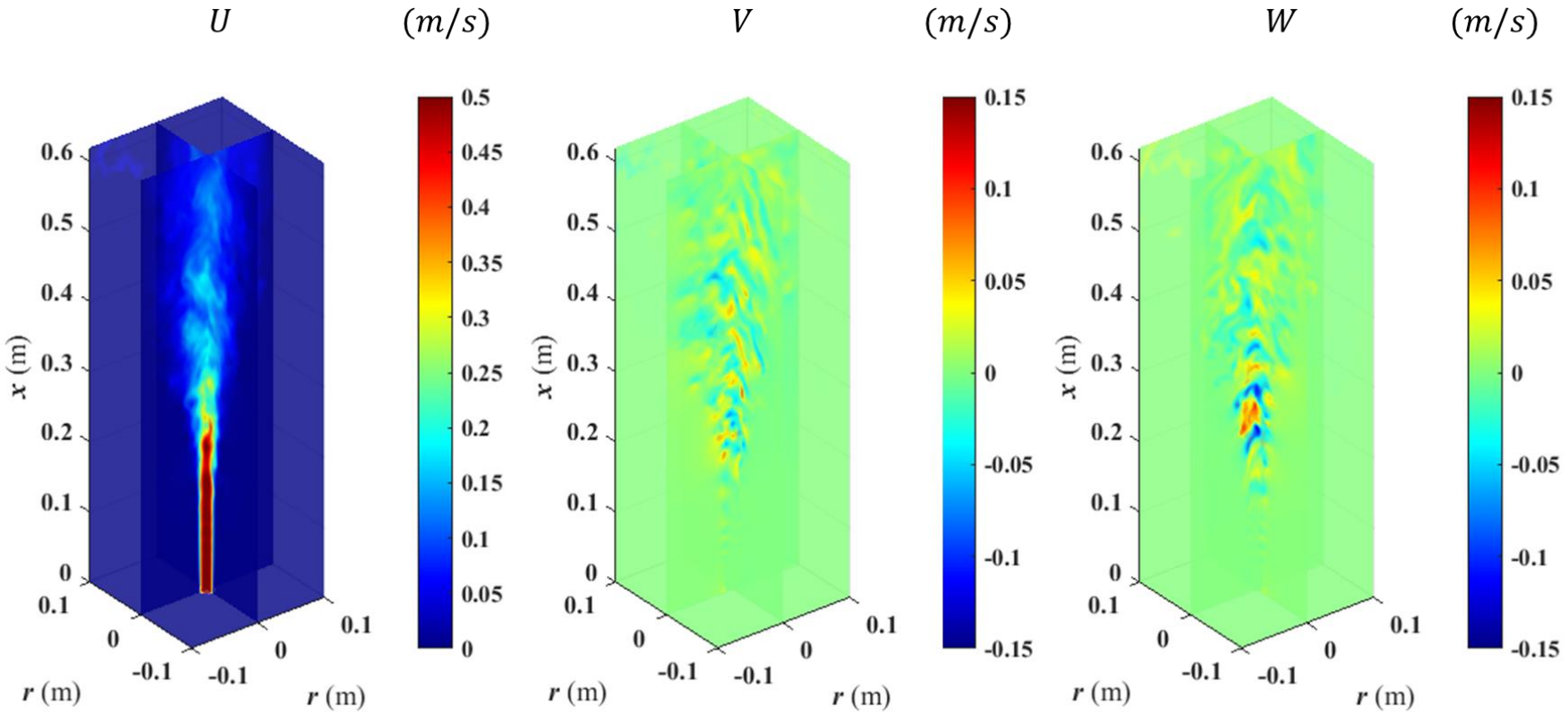

Figure 66. The three-dimensional time-averaged velocity components in the longitudinal direction (U) and radial directions (V, W) were measured in a turbulent jet at a Reynolds number of 10,000.

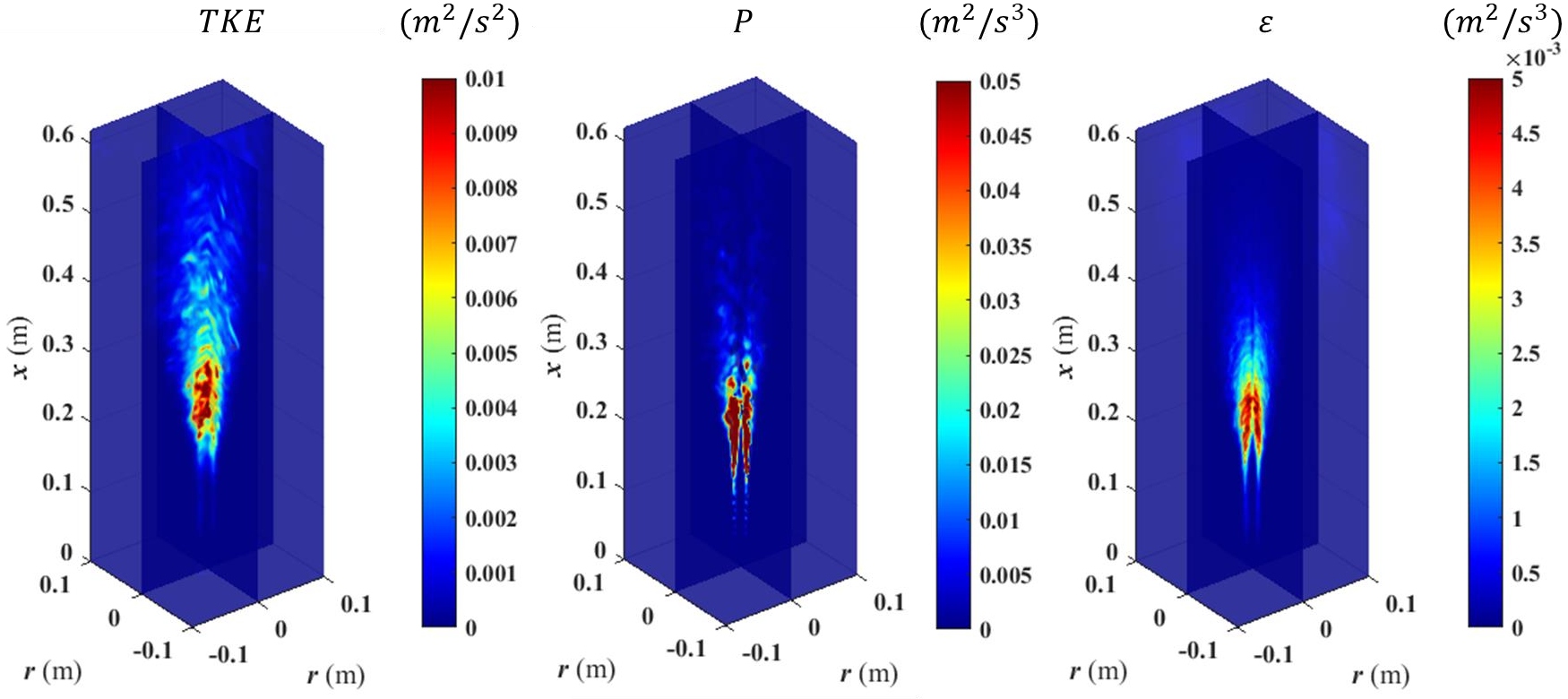

Figure 67. The three-dimensional turbulent kinetic energy (TKE), production (*P*), and dissipation (*ε*) were measured in a turbulent jet at a Reynolds number of 10,000.

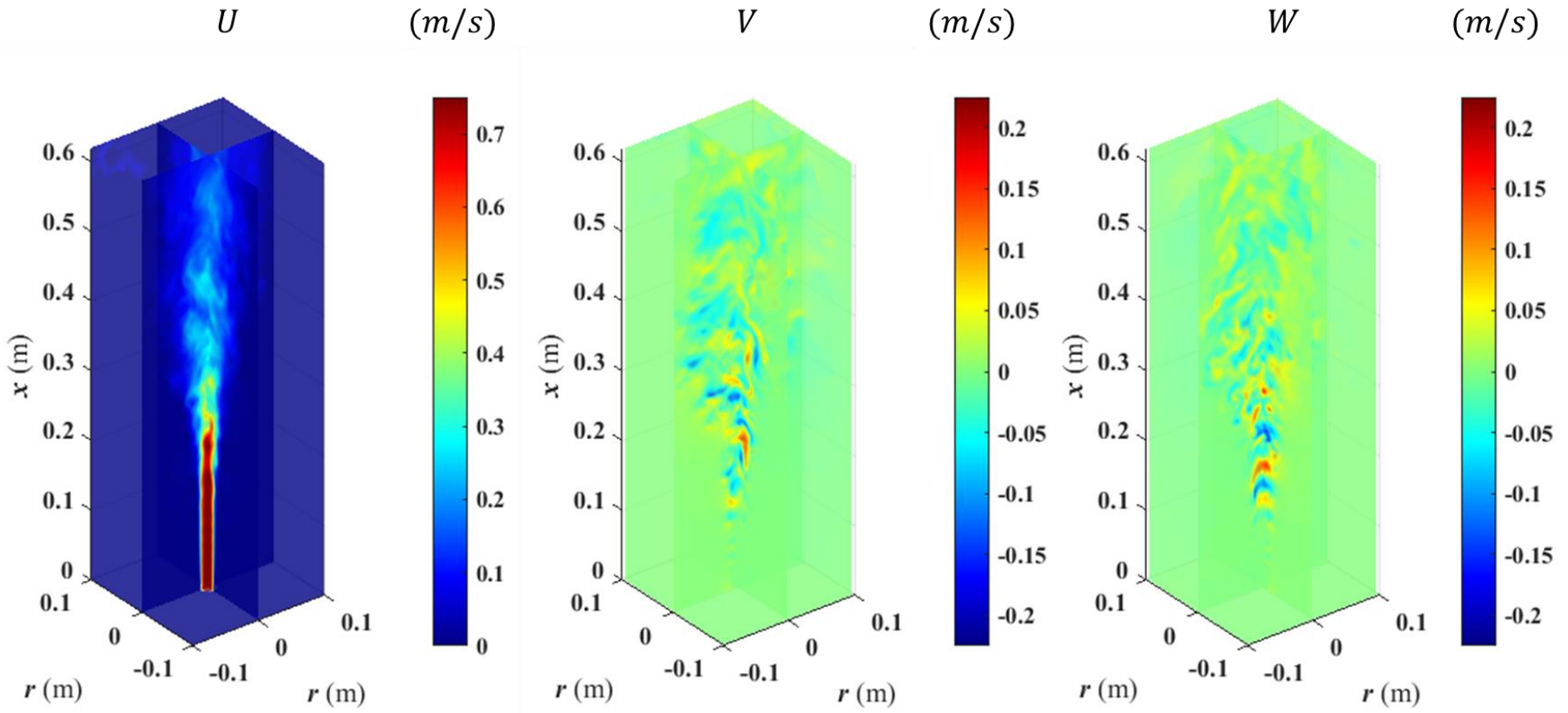

Figure 68. The three-dimensional time-averaged velocity components in the longitudinal direction (U) and radial directions (V,W) were measured in a turbulent jet at a Reynolds number of 15,000.

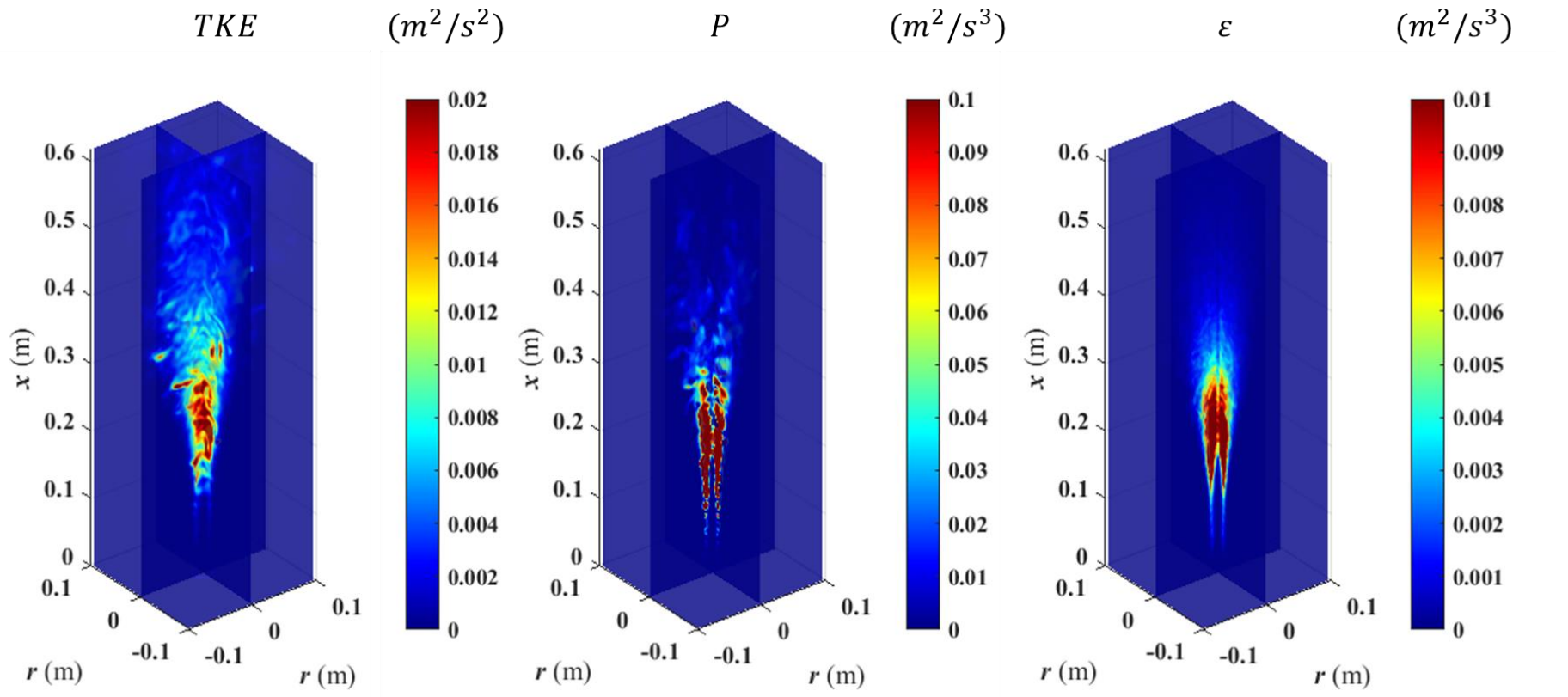

Figure 69. The three-dimensional turbulent kinetic energy (TKE), production (P), and dissipation (ε) were measured in a turbulent jet at a Reynolds number of 15,000.

Table 21 provides a summary of the NRMSE and R-squared values for TKE, production, and dissipation. The R-squared values are 0.885, 0.5165, and 0.7348, respectively. Notably, the simulation of TKE using only time-averaged velocity as input demonstrates the best performance among the three variables. On the other hand, the NRMSE values for TKE, production, and dissipation are reported as 1.40%, 0.73%, and 1.97%, respectively. According to the NRMSE values, production exhibits the best simulation performance among the three variables. The NRMSE takes into account the scale of the variables and provides a normalized measure of the error. In this case, the NRMSE values indicate that the relative error in predicting production is the smallest compared to TKE and dissipation. The lower NRMSE for production suggests that the ML model successfully captures the patterns and variations in the production variable, resulting in a more accurate simulation outcome for production compared to TKE and dissipation.

Table 21. Normalized Root Mean Square Error (NRMSE) and R-squared of Turbulent kinetic energy, production and dissipation

	NRMSE	R-squared
TKE	1.40 %	0.8850
Production	0.73 %	0.5165
dissipation	1.97 %	0.7348

Figure 70 presents the three-dimensional plots of TKE, depicting both the ground truth and the ML results. TKE exhibits its highest value near x=0.17 m and expands as the distance from the jet inlet increases. Comparing the ML result with the ground truth, the three-dimensional TKE plot of the ML result shows a similar shape but slightly lower TKE values along the centerline. Examining the cross-sectional TKE plots at specific x-values, such as x=0.344 m in Figure 71 and x=0.551 m in Figure 72, both the ground truth and the ML result demonstrate strong TKE values near r=0. This trend is captured

by the ML model at these specific locations. Notably, at x=0.551 m, the width of TKE is wider compared to that at x=0.344 m, indicating that TKE spreads more extensively as it moves away from the jet inlet. Analyzing the azimuthal-averaged TKE plot of the ML result, its width is similar to that of the ground truth. However, there are differences in the values near the centerline (where r is almost zero). Despite these differences, the R-squared value remains relatively high, suggesting an adequate agreement between the ML result and the ground truth in capturing the overall TKE characteristics.

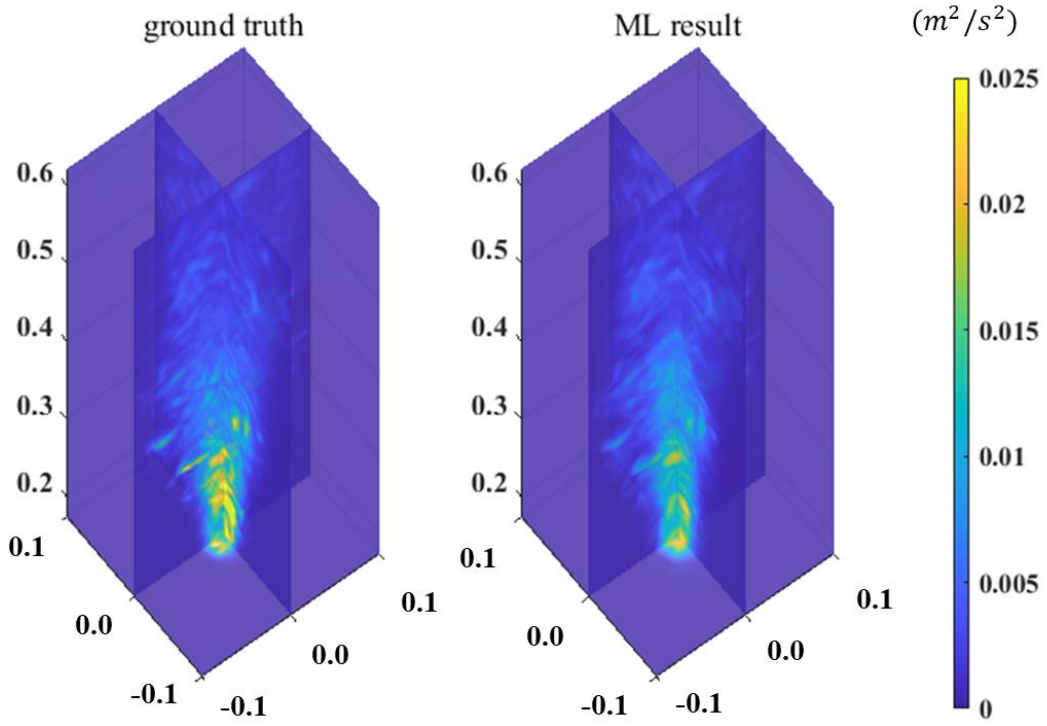

Figure 70. The three-dimensional Turbulent kinetic energy (TKE) measured in a turbulent jet at a Reynolds number of 15,000 and ML results.

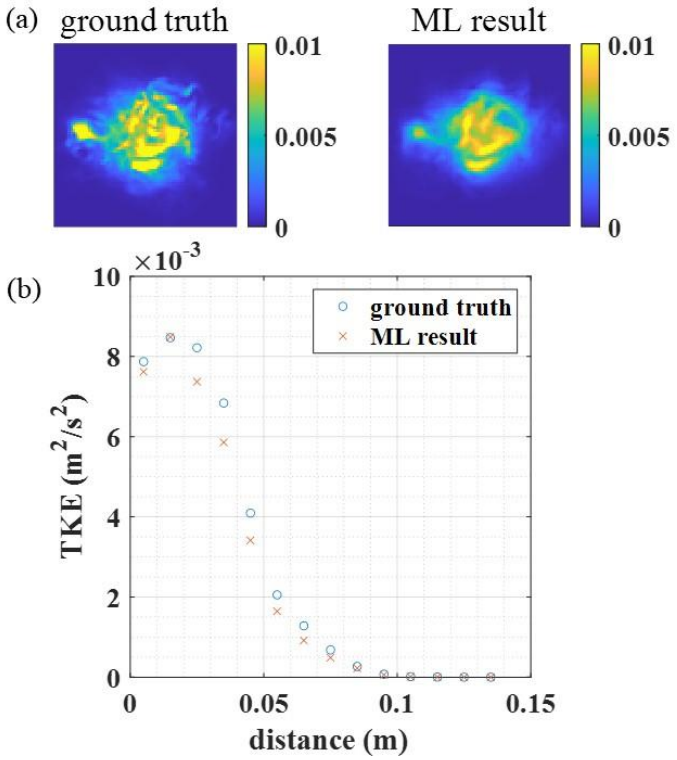

Figure 71. (a) Cross section and (b) azimuthal averaged TKE at x = 0.344 m

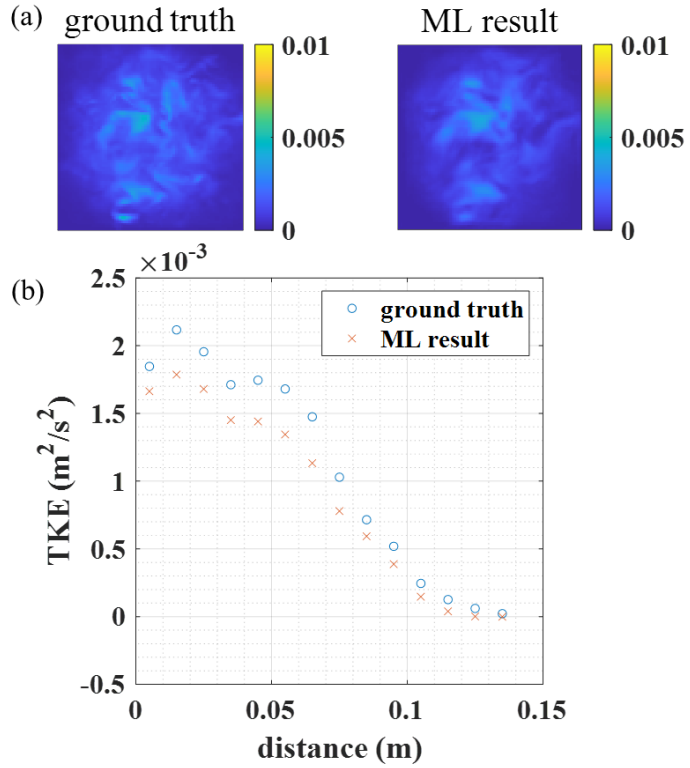

Figure 72. (a) Cross section and (b) azimuthal averaged TKE at x = 0.551 m

According to Figure 73, the three-dimensional representation of production reveals that both the ground truth and ML exhibit prominent values near the inlet, with a gradual decrease in production as the distance from the jet flow increases. As r approaches 0, positive production values are concentrated in the vicinity. The overall three-dimensional shapes of the ground truth and ML result demonstrate a notable similarity. When comparing the azimuthal-averaged plots at x=0.344 m and x=0.551 m, it is evident that the shapes of the graphs for both the ML result and the ground truth are highly similar (Figure 74, Figure 75). However, there were slight differences observed between the ML result and the ground truth in the maximum peak values at both x=0.344 m and x=0.551 m. Moreover, it can be observed that as the x value decreases, particularly closer to the inlet, the values tend to be generally larger.

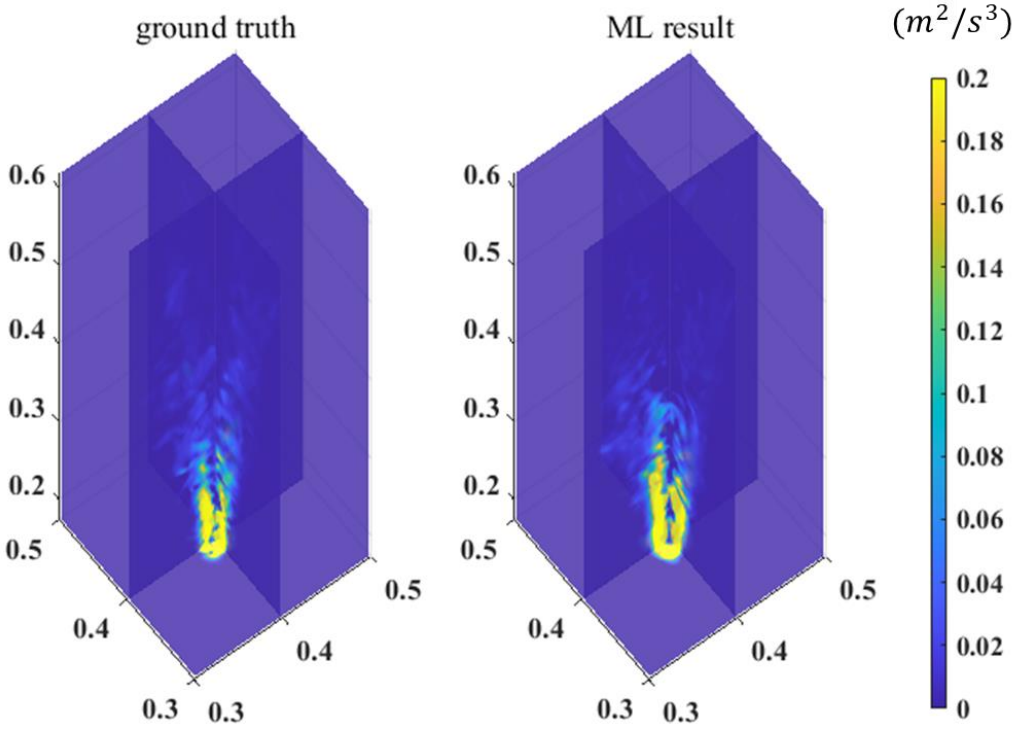

Figure 73. The three-dimensional production measured in a turbulent jet at a Reynolds number of 15,000 and ML results.

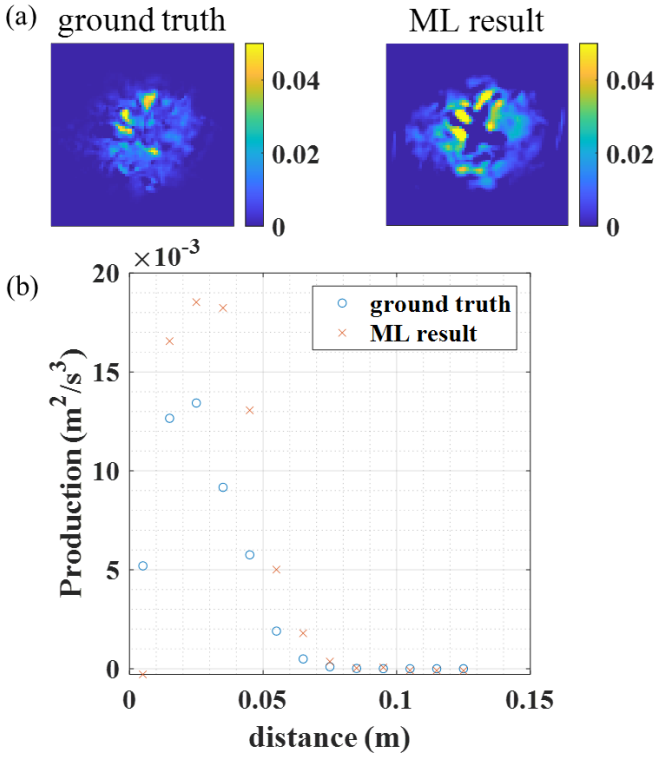

Figure 74. (a) Cross section and (b) azimuthal averaged production at x = 0.344 m

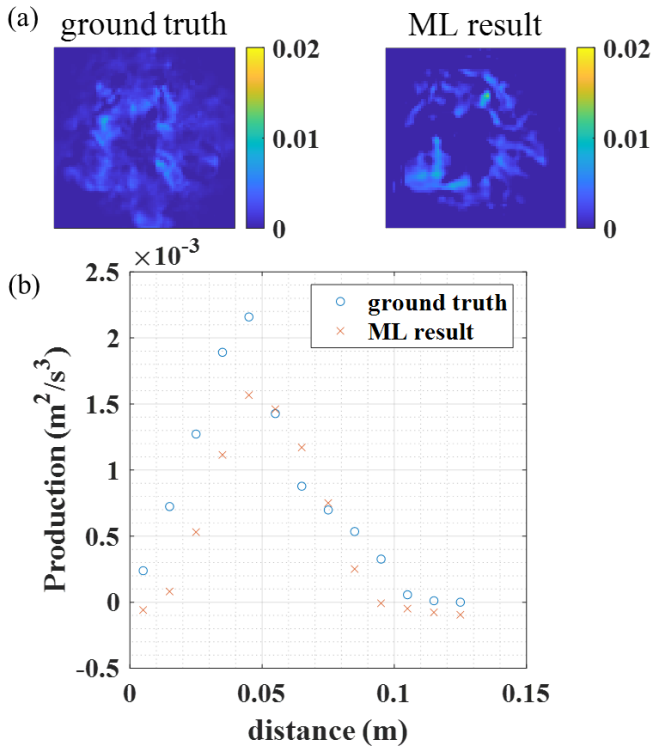

Figure 75. (a) Cross section and (b) azimuthal averaged production at x = 0.551 m

Figure 76 illustrates the dissipation distribution from x=0.176 to 0.62 m. In the ground truth, it is observed that as the radial distance (r) approaches 0, the dissipation values tend to increase. However, when r reaches exactly 0, the dissipation value becomes relatively small. This phenomenon indicates that the dissipation is high in the vicinity of r=0 but decreases abruptly when r reaches the exact center point. On the other hand, in the ML results, the clear pattern of increasing dissipation values as r approaches 0 is not as evident. The dissipation values in the ML results show a more gradual change as r decreases, and the distinction near r=0 is not as pronounced compared to the ground truth. To further explore this phenomenon, the dissipation cross-section and azimuthally-averaged result can be analyzed at x=0.344 and 0.551, respectively, as depicted in Figure 77 and Figure 78. Figure 77demonstrates a notable agreement between the ground truth and ML result. However, in the case of Figure 78, while the overall shape appears similar, there are slight discrepancies in the values obtained.

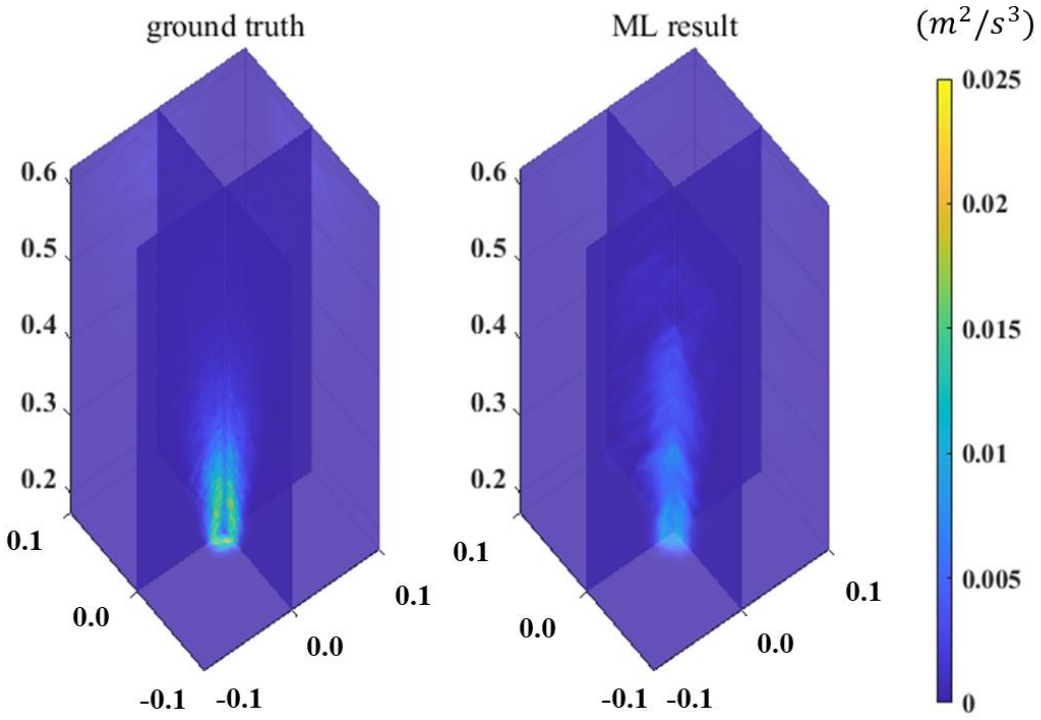

Figure 76. The three-dimensional dissipation (ε) were measured in a turbulent jet at a Reynolds number of 15,000 and ML results.

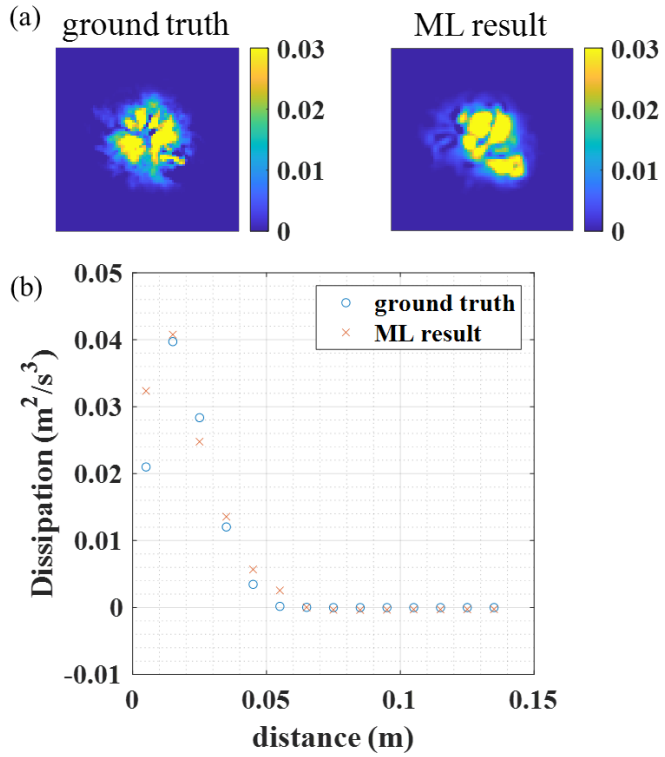

Figure 77. (a) Cross section and (b) azimuthal averaged dissipation at x = 0.344 m

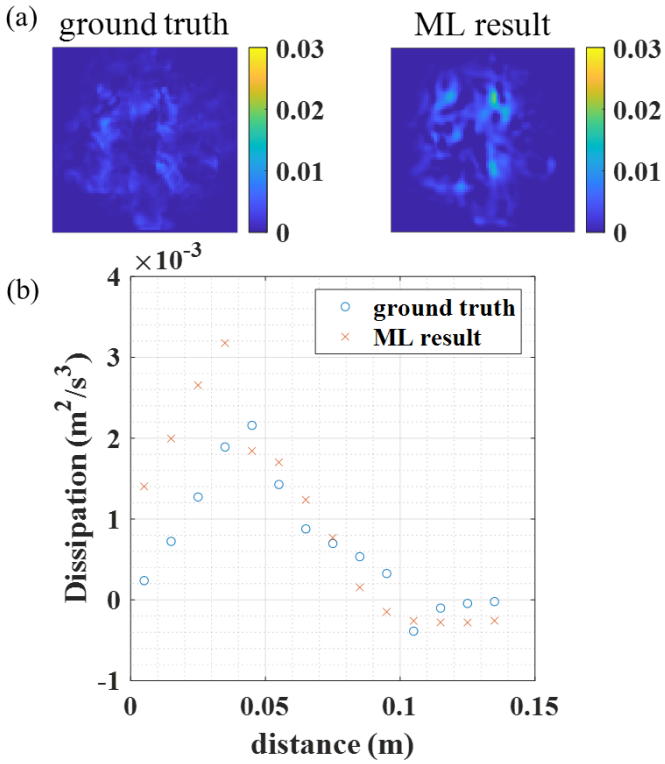

Figure 78. (a) Cross section and (b) azimuthal averaged dissipation at x = 0.551 m

CHAPTER 6. CONCLUSION

In this study, we applied the well-known CNN and GAN algorithms to accurately predict SGS turbulent flow behaviors in jet flows and evaluated their performance. Three-dimensional LES data set of such jet flows, yielded by a CFD software, OpenFOAM, is used as the ground-truth for training the aforementioned ML algorithms. The LES data set began to be collected ten minutes after the simulation started to ensure the fully developed jet flow. The good agreement between the observation data from several laboratory experiments and the simulated result of LES confirms that the LES data set can be used as a reliable ground-truth. The LES data sets, consisting of various grid sizes and Reynolds numbers, are used to train the CNN and GAN algorithms, each having two different architectures. In this study, the LES data can be considered as the output and input in the train set simultaneously. The data for the first 12 minutes are used to train the algorithms, and those for the last 8 minutes are utilized to test the algorithms. The evaluation of the performance of the trained CNN and GAN algorithms is presented, comparing them to each other. It was determined that the training times for the CNN and GAN algorithms are significantly shorter than the required times for simulating the LES (with the Smagorinsky model) for the same time period. This finding clearly shows the outstanding advantage of using the ML techniques to replicate the jet flow characteristics. The main findings of this study are summarized in the following sections.

6.1 Energy Spectrum for the Jet Flow Yielded by LES

The temporal and spatial energy spectra for the jet flow were constructed based on each axial and lateral flow velocity quantified by LES. Both temporal spectra established by the axial velocity and lateral velocity clearly show the energy-containing area and energy decay parts. It is revealed that the energy ranges of both energy spectra are almost the same, indicating the same amount of energy and

decay at the same rate along the axial and lateral directions. For the spatial energy spectra, the energy tends to decrease as the frequency increases due to the energy cascade. However, at certain frequencies, the over-rated energy was also observed, resulting from the unestablished flow structures just ejected from the inlet. When the spatial energy spectra were drawn again for the LES data in the fluid domain, excluding the first 5cm in the axial axis from the jet inlet, it was confirmed that the over-estimated energy region was alleviated. The energy range appears to be almost in line with to that of the temporal energy spectra. Based on these findings, when conducting a jet flow study, it can be deemed reasonable to select an analysis domain excluding a distance from the jet inlet to avoid overvaluing small-scale eddies.

6.2 Convolutional Neural Network

The results of the CNN algorithm were classified into four distinct outcomes based on different architectures, grid sizes of the training and testing sets, and Reynolds numbers. The overall results indicate that the first architecture of CNN performs poorly, in terms of velocity, turbulence intensity, and turbulent kinetic energy when the grid size of the LES trains set data is larger (20%) than that of the highest resolution LES train data set. On the other hand, it is confirmed that, with the second CNN architecture, the aforementioned flow properties by the CNN algorithm are well reproduced in accordance with the results of the highest resolution LES test set in spite of the larger grid size of LES train set. The R-squared value of 0.981 is the best fit with the ground truth compared to other cases. Similarly, the spatial energy spectrum for the second architecture is drawn properly in line with that for the highest resolution LES data. This good agreement can be attributed to the fact that the information of velocity fields was well preserved and remembered while passing through the convolutional blocks because two paths are located in the middle of the main path of the CNN model. The good performance of the CNN algorithm with the second architecture continues when the grid size of the LES test set is

increased (16%). Specifically, the R-squared value is 0.963, and the normalized RMSE is 0.0266 at this grid size. It is also verified that the CNN algorithm trained based on an increased (by 50 %) Reynolds number at the same grid size as the former can predict the flow velocity and turbulence intensity quite reasonably, consistently with the highest resolution LES test set.

The Taylor diagram was presented to evaluate the ability of each CNN architecture in terms of the correlation between the fluid data from the CNN algorithm and the highest resolution LES test set. The correlation when the first CNN architecture was applied is shown to be low, and the RMSE was also high. This implies that when the grid size of the input train set is somewhat larger (16 %) than that of the output train set, it may be inadequate to use the first CNN structure to predict the instantaneous velocity over time. On the other hand, it can be seen in the Taylor diagram that the data predicted by the second CNN architecture exhibited a strong correlation with the highest resolution LES test set. It is noteworthy that the correlation value did not significantly change as the grid size of the LES test set grew.

Summarizing the results above, it can be seen that the second architecture of CNN has some extrapolation capability and can learn the intrinsic mechanism based on the different grid sizes and Reynolds numbers. Consequently, it is confirmed that the three-path CNN model (the second architecture) can accurately reproduce the flow properties regardless of the grid size of the input train set and the Reynolds number considered in this study.

6.3 Generative Adversarial Network

The results of the GAN algorithm were classified into four distinct outcomes based on different architectures, grid sizes of the training and testing sets, and Reynolds numbers, which are the same conditions as the CNN model. The results of the first GAN architecture indicate that its performance in

terms of instantaneous velocity and TKE is inadequate. Specifically, the R-squared value of TKE is negative. In contrast, it is demonstrated that the flow features of the second design correspond well with the velocity field of the highest resolution LES dataset, despite the higher grid size in the input set than the output set. However, it is challenging to reproduce the TKE, with an R-squared value of only 0.194. The spatial energy spectrum for the second GAN architecture is similarly reconstructed to that of the highest resolution LES data, indicating good performance. This reasonable agreement is possible due to the super resolution GAN model's convolutional layer, which can detect small-scale flow characteristics. However, when the grid size of the test input data increases by 16%, it was found to be challenging for the GAN model with the second architecture to predict the flow behaviors. The mean velocity has an R-squared value of 0.927, whereas the TKE has an R-squared value of 0.074. These results suggest that the GAN algorithm faces difficulties in reproducing the velocity fluctuation due to the noise present in the velocity field generated by GAN.

The Taylor diagram constructed by instantaneous flow velocities indicates that the correlation and RMSE are low when applying the first GAN architecture. When the grid size of the input train set is somewhat bigger (16 %) than that of the output train set, the initial GAN structure cannot be suitable to predict the fluctuating components of velocity. In contrast, the Taylor diagram established by the data predicted by the second GAN design reveals that it has a strong correlation with the highest resolution LES test set. However, it turned out that the correlation decreases as the grid size of the LES test set grows.

Summarizing the results above suggests that the second architecture of GAN can reproduce instantaneous velocity fields, but it struggles with reproducing statistical flow quantities and spatial correlations. It was found that the noise generated by the GAN model led to velocity fluctuations that were not consistent with the ground truth. Consequently, it was confirmed that the super-resolution GAN model can accurately reproduce only the mean velocity field regardless of the grid sizes considered in this study, but its performance may be also affected by an increased grid size.

6.4 Time Averaged Turbulent Characteristics

This study investigates the accuracy of machine learning techniques in simulating turbulence characteristics using time-averaged velocity data. The input data consisted of time-averaged velocity values, and the output data are time-averaged TKE, production, and dissipation. The simulation results showed that the mean axial velocity was relatively higher than the radial velocities, and the highest production and dissipation values were located near the starting point of the shear stress region. The ML model was trained and tested using different Reynolds numbers, with the train set at the Reynolds number of 10,000 and the test set at the Reynolds number of 15,000. The normalized RMSE and R-squared values were calculated for TKE, production, and dissipation. The simulation of TKE using only time-averaged velocity as input showed the best performance among the three variables. The three-dimensional plots and cross-sectional plots of TKE, production, and dissipation showed that the machine learning model reasonably predicts the shape of the ground truth, but the values is slightly different near the centerline. Overall, the results demonstrated the potential of using machine learning techniques in accurately simulating turbulence characteristics using time-averaged velocity data. Furthermore, the results showed that the machine learning model's performance was consistent across different Reynolds numbers, indicating its robustness and applicability to reproduce the flow in a range of fluid conditions.

Reference

Abdel-Rahman, A. (2010). A review of effects of initial and boundary conditions on turbulent jets. WSEAS transactions on Fluid Mechanics, 4(5), 257-275.

Agarwal, S., Gicquel, L., Duchaine, F., Odier, N., & Dombart, J. (2021). Analysis of the Unsteady Flow Field Inside a Fan-Shaped Cooling Hole Predicted by Large Eddy Simulation. Journal of Turbomachinery, 143(3).

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., . . . Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of big Data, 8(1), 1-74.

Ashgriz, N., & Mostaghimi, J. (2002). An introduction to computational fluid dynamics. Fluid flow handbook, 1, 1-49.

Ball, C., Fellouah, H., & Pollard, A. (2012). The flow field in turbulent round free jets. Progress in aerospace sciences, 50, 1-26.

Bayraktar, S., & Yilmaz, T. (2011). Experimental analysis of transverse jet using various decomposition techniques. Journal of mechanical science and technology, 25(5), 1325-1333.

Bogey, C., & Bailly, C. (2005). Effects of inflow conditions and forcing on subsonic jet flows and noise. AIAA journal, 43(5), 1000-1007.

Burger, W., & Burge, M. J. (2016). Digital image processing: an algorithmic introduction using Java: Springer.

Chatterjee, A. (2000). An introduction to the proper orthogonal decomposition. Current science, 808-817.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018).

Generative adversarial networks: An overview. IEEE signal processing magazine, 35(1), 53-65.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., . . . Yang, K. (2012). Large scale distributed deep networks. Advances in neural information processing systems, 25.

Dejoan, A., & Leschziner, M. (2005). Large eddy simulation of a plane turbulent wall jet. Physics of Fluids, 17(2), 025102.

Deng, Z., He, C., Liu, Y., & Kim, K. C. (2019). Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework. Physics of Fluids, 31(12), 125111.

Deo, R. C., Mi, J., & Nathan, G. J. (2008). The influence of Reynolds number on a plane jet. Physics of Fluids, 20(7), 075108.

Dogra, A., & Bhalla, P. (2014). Image sharpening by gaussian and butterworth high pass filter. Biomedical and pharmacology journal, 7(2), 707-713.

Dong, C., Loy, C. C., He, K., & Tang, X. (2015). Image super-resolution using deep convolutional networks. IEEE transactions on pattern analysis and machine intelligence, 38(2), 295-307.

Dubey, V. R. (2014). Quaternion Fourier transform for colour images. International Journal of Computer Science and Information Technologies, 5(3), 4411-4416.

El Naqa, I., & Murphy, M. J. (2015). What is machine learning? In machine learning in radiation oncology (pp. 3-11): Springer.

Erdil, A., Kodal, A., & Aydin, K. (2002). Decomposition of turbulent velocity fields in an SI engine. Flow, turbulence and combustion, 68(2), 91-110.

Faghani, E., Saemi, S. D., Maddahian, R., & Farhanieh, B. (2011). On the effect of inflow conditions in simulation of a turbulent round jet. Archive of Applied Mechanics, 81(10), 1439-1453.

Fang, S., Chen, Y., Xu, Z., Otoo, E., & Lu, S. (2019). An improved integral model for a non-buoyant turbulent jet in wave environment. Water, 11(4), 765.

Ferziger, J. H., Perić, M., & Street, R. L. (2002). Computational methods for fluid dynamics (Vol. 3): Springer.

Fischer, H., List, E., Koh, R., Imberger, J., & Brooks, N. (1979). Mixing in inland and Coastal waters Academic Press 1979.

Galdi, G. (2011). An introduction to the mathematical theory of the Navier-Stokes equations: Steady-state problems: Springer Science & Business Media.

Gamahara, M., & Hattori, Y. (2017). Searching for turbulence models by artificial neural network. Physical Review Fluids, 2(5), 054604.

Georgiadis, N. J., Yoder, D. A., & Engblom, W. A. (2006). Evaluation of modified two-equation turbulence models for jet flow predictions. AIAA journal, 44(12), 3107-3114.

Germano, M., Piomelli, U., Moin, P., & Cabot, W. H. (1991). A dynamic subgrid-scale eddy viscosity model. Physics of Fluids A: Fluid Dynamics, 3(7), 1760-1765.

Gohil, T. B., Saha, A. K., & Muralidhar, K. (2014). Large eddy simulation of a free circular jet. Journal of Fluids Engineering, 136(5).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning: MIT press.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial nets (Advances in neural information processing systems)(pp. 2672–2680). Red Hook, NY Curran.

Gourlay, M. J., Arendt, S., Fritts, D., & Werne, J. (2001). Numerical modeling of initially turbulent wakes with net momentum. Physics of Fluids, 13(12), 3783-3802.

Greenshields, C. J. (2015). OpenFOAM user guide. OpenFOAM Foundation Ltd, version, 3(1), 47.

Hefny, M. M., & Ooka, R. (2009). CFD analysis of pollutant dispersion around buildings: Effect of cell geometry. Building and Environment, 44(8), 1699-1706.

Helm, J. M., Swiergosz, A. M., Haeberle, H. S., Karnuta, J. M., Schaffer, J. L., Krebs, V. E., . . . Ramkumar, P. N. (2020). Machine learning and artificial intelligence: definitions, applications, and future directions. Current reviews in musculoskeletal medicine, 13(1), 69-76.

Ho, Y., & Wookey, S. (2019). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. IEEE Access, 8, 4806-4813.

Hoffman, F. (1997). An introduction to Fourier theory. Extraído el, 2.

Huang, X., Wang, L.-l., & Xu, J. (2020). Numerical Study on Dynamical Structures and the Destratification of Vertical Turbulent Jets in Stratified Environment. Water, 12(8), 2085.

Hurtik, P., Tomasiello, S., Hula, J., & Hynar, D. (2022). Binary cross-entropy with dynamical clipping. Neural Computing and Applications, 1-13.

Jadon, S. (2020). A survey of loss functions for semantic segmentation. Paper presented at the 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB).

Jiang, X., & Lai, C.-H. (2016). Numerical techniques for direct and large-eddy simulations: CRC press.

Jin, X., Cheng, P., Chen, W.-L., & Li, H. (2018). 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, 30(4), 047105.

Johansson, P. B., George, W. K., & Gourlay, M. J. (2003). Equilibrium similarity, effects of initial

conditions and local Reynolds number on the axisymmetric wake. Physics of Fluids, 15(3), 603-617.

Karasu, İ. (2020). Flow control over a diamond-shaped cylinder using slits. Experimental Thermal and Fluid Science, 112, 109992.

Khorsandi, B., Gaskin, S., & Mydlarski, L. (2013). Effect of background turbulence on an axisymmetric turbulent jet. Journal of Fluid Mechanics, 736, 250-286.

Kim, H., Kim, J., Won, S., & Lee, C. (2021). Unsupervised deep learning for super-resolution reconstruction of turbulence. Journal of Fluid Mechanics, 910.

Kim, J., & Choi, H. (2009). Large eddy simulation of a circular jet: effect of inflow conditions on the near field. Journal of Fluid Mechanics, 620, 383-411.

Kolmogorov, A. N. (1991). The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences, 434(1890), 9-13.

Kumar, M., Tiwari, N. K., & Ranjan, S. (2022). Application of Machine Learning Methods in Estimating the Oxygenation Performance of Various Configurations of Plunging Hollow Jet Aerators. Journal of Environmental Engineering, 148(11), 04022070.

Kwon, S. J., & Seo, I. W. (2005). Reynolds number effects on the behavior of a non-buoyant round jet. Experiments in fluids, 38(6), 801-812.

Lee, J. H.-w., Chu, V., & Chu, V. H. (2003). Turbulent jets and plumes: a Lagrangian approach (Vol. 1): Springer Science & Business Media.

Lesieur, M. (1987). Turbulence in fluids: stochastic and numerical modelling (Vol. 488): Nijhoff Boston, MA.

Liepmann, H. W., & Laufer, J. (1947). Investigations of free turbulent mixing. Retrieved from

Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. Journal of Fluid Mechanics, 807, 155-166.

List, E. J. (1982). Turbulent jets and plumes. Annual review of fluid mechanics, 14(1), 189-212.

Loshchilov, I., & Hutter, F. (2018). Fixing weight decay regularization in adam.

Matsuda, T., & Sakakibara, J. (2005). On the vortical structure in a round jet. Physics of Fluids, 17(2), 025106.

Maulik, R., & San, O. (2017). A neural network approach for the blind deconvolution of turbulent flows. Journal of Fluid Mechanics, 831, 151-181.

Moeini, M., Khorsandi, B., & Mydlarski, L. (2020). Effect of acoustic Doppler velocimetry sampling frequency on statistical measurements of turbulent axisymmetric jets. Journal of Hydraulic Engineering, 146(7), 04020048.

Mossa, M., & Davies, P. A. (2018). Some aspects of turbulent mixing of jets in the marine environment. Water, 10(4), 522.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

O'Neill, P., Soria, J., & Honnery, D. (2004). The stability of low Reynolds number round jets. Experiments in fluids, 36(3), 473-483.

Oh, S.-H., Mizutani, N., & Suh, K.-D. (2007). Investigation of decomposition methods of turbulent flow field beneath wind waves. In Coastal Engineering 2006: (In 5 Volumes) (pp. 251-260): World Scientific.

Ongsulee, P. (2017). Artificial intelligence, machine learning and deep learning. Paper presented at the 2017 15th international conference on ICT and knowledge engineering (ICT&KE).

Panchapakesan, N. R., & Lumley, J. L. (1993). Turbulence measurements in axisymmetric jets of air and helium. Part 1. Air jet. Journal of Fluid Mechanics, 246, 197-223.

Pannu, A. (2015). Artificial intelligence and its application in different areas. Artificial Intelligence, 4(10), 79-84.

Park, J., & Choi, H. (2021). Toward neural-network-based large eddy simulation: Application to turbulent channel flow. Journal of Fluid Mechanics, 914.

Pawar, S., San, O., Rasheed, A., & Vedula, P. (2020). A priori analysis on deep learning of subgrid-scale parameterizations for Kraichnan turbulence. Theoretical and Computational Fluid Dynamics, 34(4), 429-455.

Peiró, J., & Sherwin, S. (2005). Finite difference, finite element and finite volume methods for partial differential equations. In Handbook of materials modeling (pp. 2415-2446): Springer.

Piomelli, U. (1999). Large-eddy simulation: achievements and challenges. Progress in aerospace sciences, 35(4), 335-362.

Pope, S. B., & Pope, S. B. (2000). Turbulent flows: Cambridge university press.

Rahman, M. A., Muniyandi, R. C., Islam, K. T., & Rahman, M. M. (2019). Ovarian Cancer Classification Accuracy Analysis Using 15-Neuron Artificial Neural Networks Model. Paper presented at the 2019 IEEE Student Conference on Research and Development (SCOReD).

Raissi, M., Yazdani, A., & Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science, 367(6481), 1026-1030.

Revuelta, A., Sánchez, A. L., & Linán, A. (2002). The virtual origin as a first-order correction for the far-field description of laminar jets. Physics of Fluids, 14(6), 1821-1824.

Richardson, L. F. (1922). Weather prediction by numerical process: University Press.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Rzeszotarski, M., Royer, F., & Gilmore, G. (1983). Introduction to two-dimensional Fourier analysis. Behavior Research Methods & Instrumentation, 15(2), 308-318.

Sagaut, P. (2005). Large eddy simulation for incompressible flows: an introduction: Springer Science & Business Media.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. towards data science, 6(12), 310-316.

Shinneeb, A.-M., Bugg, J., & Balachandar, R. (2008). Quantitative investigation of vortical structures in the near-exit region of an axisymmetric turbulent jet. Journal of Turbulence(9), N19.

Starn, J. (2001). A simple fluid solver based on the FFT. Journal of graphics tools, 6(2), 43-52.

Taub, G., Lee, H., Balachandar, S., & Sherif, S. (2013). A direct numerical simulation study of higher order statistics in a turbulent round jet. Physics of Fluids, 25(11), 115102.

Tran, P. T. (2019). On the convergence proof of amsgrad and a new version. IEEE Access, 7, 61706-61716.

Uddin, M., & Pollard, A. (2007). Self-similarity of coflowing jets: the virtual origin. Physics of Fluids, 19(6), 068103.

Wang, C., Barghi, S., & Zhu, J. (2014). Hydrodynamics and reactor performance evaluation of a high flux gas-solids circulating fluidized bed downer: Experimental study. AIChE Journal, 60(10), 3412-3423.

Wang, Q., Ma, Y., Zhao, K., & Tian, Y. (2022). A comprehensive survey of loss functions in machine learning. Annals of Data Science, 9(2), 187-212.

Wang, Z., He, P., Lv, Y., Zhou, J., Fan, J., & Cen, K. (2010). Direct numerical simulation of subsonic round turbulent jet. Flow, turbulence and combustion, 84(4), 669-686.

Weaver, D. S., & Mišković, S. (2021). A Study of RANS Turbulence Models in Fully Turbulent Jets: A Perspective for CFD-DEM Simulations. Fluids, 6(8), 271.

Wu, J.-L., Xiao, H., & Paterson, E. (2018). Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. Physical Review Fluids, 3(7), 074602.

Xiao, S., Peng, C., & Yang, D. (2021). Large-eddy simulation of bubble plume in stratified crossflow. Physical Review Fluids, 6(4), 044613.

Xu, G., & Antonia, R. (2002). Effect of different initial conditions on a turbulent round free jet. Experiments in fluids, 33(5), 677-683.

Yılmaz, T., & Kodal, A. (2000). An investigation of forced structures in turbulent jet flows. Experiments in fluids, 29(6), 564-572.

Zupan, J. (1994). Introduction to artificial neural network (ANN) methods: what they are and how to use them. Acta Chimica Slovenica, 41, 327-327.

초 록

본 연구에서는 합성곱 신경망과 생성적 적대 신경망을 활용하여 제트 흐름에서 유속과 난류 특성을 생성하는 연구를 진행하였다. 전산유체역학 소프트웨어인 오픈폼을 활용해 원형 난류 제트 흐름의 3차원 큰에디모사 데이터를 도출하여 기계 학습 알고리즘을 학습시키기 위한 참값으로 사용하였다. 수치모의 결과와 실험실 실험 데이터가 적절하게 일치하는 것으로 확인되었으며, 이에 따라 큰에디모사의 신뢰성이 확인되었다. 본 연구에서는 큰에디모사의 아격자 부분을 제외한 해상 된(resolved) 유속자료를 입력 데이터로 사용하였다. 이후, 기계 학습 모델을 활용하여 아격자 요소를 포함한 유속장을 생성하였다. 본 연구에서는 입력 데이터를 사용하여 아격자 정보를 예측하고 정확한 유체 흐름 특성을 담은 완전한 유속장을 생성하였다. 다양한 그리드 크기와 레이놀즈수를 사용하여 합성곱 신경망과 생성적 적대 신경망 알고리즘의 성능을 평가하였다.

합성곱 신경망 알고리즘은 두 가지 아키텍처로 구성되었다. 첫 번째 아키텍처는 층이 한 경로로 연결된 구조이고, 두 번째 아키텍처는 층이 세 개의 경로로 연결된 구조이다. 두 번째 아키텍처가 유속과 난류 운동 에너지 생성 측면에서 첫 번째 아키텍처보다 성능이 뛰어난 것으로 확인되었다. 또한 두 번째 아키텍처는 외삽 능력이 우수하여, 이 연구에서 고려된 입력 데이터의 그리드 크기와 레이놀즈수와 상관없이 정확한 유체 흐름 특성을 재현하는 능력을 보였다. 마찬가지로 생성적 적대 신경망 알고리즘도 두 가지 아키텍처로 구성되었다. 첫 번째 아키텍처는 완전 연결 층으로 구성되고, 두 번째 아키텍처는 합성곱 층으로 구성되었다. 두 번째 아키텍처가 유속장 재생산 측면에서 첫 번째 아키텍처보다 뛰어난 성능을 보였다. 그러나 두 생성적 적대 신경망 아키텍처 모두 합성곱 신경망에 비해 난류 특성과 공간적 상관관계를 재현하는 데 상대적으로 다소 부정확한

결과를 보였다.

또한, 본 연구에서는 합성곱 신경망 기법을 사용하여 시간 평균 된 유속 데이터를 통해 난류 특성을 재현하는 연구를 진행하였다. 입력 데이터는 시간 평균한 유속이고, 출력 데이터로는 난류 운동 에너지, 시간 평균한 생성 및 소산이 예측됐다. 합성곱 신경망 모델을 평가하기 위해 합성곱 신경망 학습에 사용되지 않은 다른 레이놀즈수를 사용하여 난류 운동 에너지, 시간 평균한 생성, 소산을 생성했다. 그 결과 합성곱 신경망 모델은 난류 특성을 적절하게 구현하는 것으로 나타났다. 결과적으로, 합성곱 신경망 모델은 시간 평균한 유속 데이터를 사용하여 난류 특성을 예측하는 데 잠재력을 가졌으며, 복잡한 난류 흐름 현상을 재현하는 적절한 기계학습 알고리즘을 선택하는 데 유용한 통찰력을 제공한다.

주요어: 난류 제트; 합성곱 신경망; 생성적 적대 신경망; 큰에디모사; 오픈폼

학 번: 2017-24760

감사의 글

박사과정을 진행하면서 저의 학위논문이 잘 마무리될 수 있게 많은 분의 도움이 있었습니다. 이 글을 통해 감사 인사를 드리고자 합니다.

우선, 박사과정 동안 큰 가르침을 주신 황진환 교수님께 진심으로 감사드립니다. 교수님께서 보여주신 학문에 대해 열린 마음과 열정을 보며 학문을 대하는 자세를 배울 수 있었습니다. 논문을 체계적으로 다듬어갈 수 있도록 짚어주신 김열우 교수님께도 감사드립니다. 또한, 유익한 가르침을 주시고 진심 어린 조언을 해주신 박용성, 손상영, 박인환 교수님 감사드립니다.

학교생활을 하면서 의지가 되어준 같은 연구실 구성원인 김남훈, 김동현, 김보경, 박형철, 손석민, 유호정, 이관호, 이창희, 정원정, 정재영, 한지수, 홍성수에게도 감사드립니다. 학업과 학위 과정도 의미가 있었지만, 이렇게 좋은 연구실원들을 만날 수 있었던 것 또한 제게 감사한 일입니다. 그리고 입학 동기로서 학위 과정 동안 어려움이 있을 때마다 함께 고민해 준 이정후 박사에게도 고마움을 전합니다.

사랑하는 제 가족들에게 감사를 전합니다. 항상 배움의 자세를 놓지 않고 본이 되어주시는 아버지, 어려운 일이 있어 낙담할 때마다 큰 위로가 되어주신 어머니께 감사드립니다. 그리고 항상 의지가 되는 지은 언니, 늘 응원해주며 곁에 있어 준 유나 언니에게 감사드립니다.

이 감사의 글에 다 담지는 못하였지만, 박사 과정을 하는 6년 동안 힘이 되어준 가족, 친구들에게 감사를 전하고 싶습니다.