Generalization of Urban Wind Field Using Fourier Neural Operators Across Different Wind Directions and Cities

Cheng Chen

A Thesis
In the Department
of
Computer Science & Software Engineering

Presented in Partial Fulfillment of the
Requirements for the Degree of
Master of Computer Science (Computer Science & Software Engineering)
at Concordia University
Montreal, Quebec, Canada

September 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Cheng Chen

Entitled: Generalization of Urban Wind Field Using Fourier Neural Operators Across Different Wind Directions and Cities

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science & Software Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

Dr. Shin Hwei Tan Chair

Dr. Shin Hwei Tan Examiner

Dr. Biao Li Examiner

Dr. Jinqiu Yang Co-supervisor

Dr. Liangzhu (Leon) Wang Co-supervisor

Dr. David Vidal Co-supervisor

Approved by _____

Dr. Joey Paquet, Chair
Department of Computer Science & Software Engineering

_____

Dr. Mourad Debbabi, Dean
Faculty of Engineering & Computer Science

Abstract

Generalization of Urban Wind Field Using Fourier Neural Operators Across Different Wind Directions and Cities

Cheng Chen

In urban environments, the most common forms of air transportation are helicopters and unmanned aerial vehicles (UAVs). There is a high demand for air transport of small and medium sized aircraft, including UAVs. Wind field simulations in urban environments are typically performed using computational fluid dynamics (CFD), and most of these models fall into the categories of direct numerical simulation (DNS) and large eddy simulation (LES). Although these models are accurate, they are time-consuming, so there is a need to develop a more convenient method to replace the traditional CFD methods. In recent years, with the rapid development of artificial intelligence technology and graphics processing unit (GPU) hardware, a promising research direction has emerged. Currently, many studies are using artificial intelligence-based deep learning techniques to transform the computational processes associated with wind field simulation. The goal of these studies is not only to achieve the accuracy of traditional CFD models, but to surpass them while significantly accelerating the computational process. In this paper, we apply the Fourier Neural Operator (FNO) method based on deep learning technology to simulate the wind field in a two-dimensional urban environment. The method uses a Fourier module to extract and learn features in the Fourier frequency domain of the input data. Compared to traditional convolutional neural network (CNN) modules, Fourier modules aim to learn global features in the Fourier frequency domain of the input data. In contrast, a convolutional neural network (CNN) module performs feature learning in the local spatial domain of the input features. In addition, the input features are processed by a Multi-Layer Perceptron (MLP) module, and the feature output of the MLP module is added to the feature output of the Fourier module. This structure is based on a residual network (ResNet), which can mitigate the phenomenon of gradient vanishing or gradient explosion that occurs when input data propagates through a multilayer network.

The FNO model ultimately maps the input features (i.e., the input wind field) to the desired output features (i.e., the output wind field dimensions). Gradients are updated through back propagation to reduce the discrepancy between the FNO model's output wind field and the actual wind field, thus facilitating the deep learning process. After a series of experiments, the optimal settings for the Fourier layer number and the intermediate feature dimensions of the MLP in the FNO model were determined. In this context, "intermediate feature dimensions" refers to the number of features extracted by the MLP module. These settings ensure that the FNO model achieves the best results on the dataset while minimizing computational overhead and resource consumption. The training phase utilized wind field data from Niigata with westerly winds, with a time step of 0.1 seconds, and the output consisted of wind fields at the same location with a time step of 1 second (i.e. 10 time steps). Experimental results demonstrated that the FNO model could predict the wind field over the entire Niigata urban area for the next 7 seconds (i.e. 70 time

steps), with an average absolute error of less than 0.5 m/s. Importantly, the FNO exhibited strong generalization capabilities in different wind conditions: although the training data consisted of westerly wind data from Niigata, the model performed well in tests with northerly winds. Further validation across different urban geometries revealed that the FNO model could accurately predict 70 time steps (7 seconds) of wind fields in the vertically flipped version of Niigata, indicating that it generalizes well when the geometry is similar to the training data. However, in Montreal, which has a significantly different urban geometry, the model's accuracy diminished after 10 time steps. This highlights the significance of urban geometry in wind field prediction. During this process, the FNO's wind field simulation was 300 times faster than that of the CityFFD model we employed, with CityFFD requiring 2.2 seconds per step, whereas FNO took only 0.006 seconds. This further underscores the potential of the FNO model for practical applications in wind field simulation.

Although it is premature to use FNO directly to replace wind field simulation due to the exponential growth of errors with time, it is possible to use it in conjunction with CityFFD and other technologies as a complementary model. For example, the wind field output by CityFFD at a given time step can be used as input to FNO, which can generate the wind field in the same area at subsequent time steps. The final output wind field can be used as input to CityFFD, thus reducing the intermediate computation time of CityFFD.

Content

List of Figures

List of Tables

List of Acronyms and Abbreviations

Acroym/Abbreviation	Full name
CNN	Convolutional Neural Network
FNO	Fourier Neural Operator
PMM	Porous Media Model
MM	Multilezone Med
FFD	Fast Fluid Dynamics
SSIM	Structure Similarity

Chapter 1 Context

1.1 Urban Wind Field Simulation and Challenges

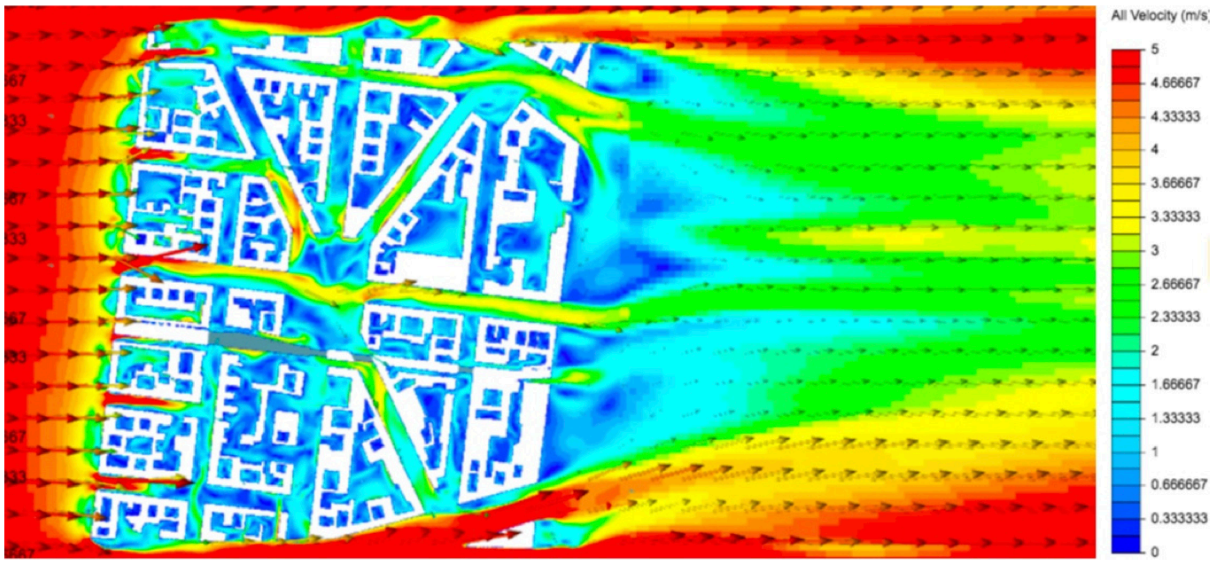

Figure 1.1 CFD wind simulation by SimScale [66] .

In recent years, the primary means of air transportation in human cities, namely helicopters and drones, have been the subject of considerable attention. The combination of power systems, distributed propulsion technologies, and new demands for commercial or medical delivery, aerial surveillance, etc., have provided established original equipment manufacturers (OEMs) such as Bell and small companies such as Beta with new solutions for urban air traffic. Nevertheless, guaranteeing the seamless operation of these novel aircraft in an urban air environment represents a formidable challenge. To address this challenge, it is necessary to conduct sufficient training in urban wind field simulations, a process that is currently complex and resource-intensive. The prevailing experimental techniques for simulating urban wind fields are wind tunnel experiments and computer simulation experiments. The high cost of wind tunnel experiments has made them a significant financial burden, leading to the adoption of computational fluid dynamics (CFD) methods for simulating urban wind fields. In urban environments, CFD methods can calculate physical variables, including but not limited to wind fields, temperature, and pressure fields. In virtual environments simulating aircraft operations, the wind field has the most significant impact on the performance of the aircraft. To accurately simulate wind field variations in urban environments, it is crucial to understand the fundamental principles governing airflow dynamics. CFD methods are grounded in these physical laws, offering strong interpretability and a solid mathematical and physical foundation for urban wind field simulations. Numerous organizations, including CAE, have employed CFD methods to

create virtual urban wind field environments for pilot training to meet new airworthiness standards.

However, despite advancements in traditional CFD software enabling the simulation of micro-urban wind fields on personal computers, high-resolution CFD simulations still demand substantial computational resources and time, posing challenges for their practical application in industrial settings. To address these limitations, researchers are exploring new methods. With the development of GPU hardware and artificial intelligence (AI) technologies, AI has provided faster and more efficient solutions for high-precision wind field simulations. This not only offers significant potential for urban planning and environmental management but also provides new approaches to overcome the limitations of traditional CFD methods.

1.2 Motivation and Objective

In order to enhance the efficiency of wind field simulation using traditional CFD methods while reducing costs, this paper draws upon deep convolutional network technology after an in-depth examination of existing CFD numerical methods and deep learning-based approaches. Among AI methods for solving partial differential equations (PDEs), such as PINN and neural operators, we selected the Fourier Neural Operator (FNO) due to its superior performance in recent AI methods. FNO achieves the highest prediction accuracy and the shortest training time while consuming the same hardware resources, making it particularly successful in optimizing micro-urban wind field simulation tasks. This method is combined with the CityFFD simulation method proposed by Mortezazadeh, M[37] to enhance the efficiency and accuracy of the simulation process. This paper examines the generalization ability and computational efficiency of the FNO method for different wind directions, urban patterns and heights. It combines the powerful ability of deep learning to identify implicit connections between data with the simulation ability of CFD, which is supported by explicit mathematical logic. The aim is to enhance the efficiency of CFD simulation and the generalization ability for different urban wind field datasets. In this paper, we utilize the CityFFD solver to generate Niigata westerly wind data as a training set and Niigata northerly wind data as a test set. These data sets are employed to assess and analyze the performance and analytical conclusions of the FNO method, and to summarize the primary contributions and innovations of this paper.

1) The FNO Fourier neural operator method is employed to generalize wind field prediction over different wind directions and urban structures. Firstly, we proposed a feature representation method for wind flow field data to address the issue of how to represent the input data for deep learning of the FNO method. We constructed a training dataset and test dataset for Niigata's unsteady-to-steady wind field using a CityFFD-based numerical simulation solver. Secondly, we conducted an experimental study on the optimal The hyper-parameters for FNO learning of the wind field, including the layer number, batch size, input-output strategy, and other model parameters, were tested against each other. In terms of performance evaluation metrics, given the primary focus on simulation accuracy in the wind task, only the average absolute error was considered, rather than the relative error. In the analysis of the dataset, we employed the spectral analysis method to elucidate the learning effect of the dataset. Ultimately, we attained an average absolute error of less than 0.5 m in the initial 70 steps (0.1 second interval for each time

step) of the output on the test set, which has already reached the simulation error level of the mainstream CFD software [64]. Moreover, each step of the prediction process is completed in a mere 0.006 seconds, a figure that is 300 times faster than that achievable with CFD solvers.

2) We propose a novel training method for fluid data in the context of deep learning. When using the complete wind field data directly for GPU training, the required number of modes exceeds the standard specifications of our GPU hardware (32GB VRAM). Conversely, if the number of modes is too low, higher frequencies, which represent more complex features in urban wind fields, cannot be captured. To solve this issue, we divided the wind field data into discrete blocks for training, enabling the capture of higher frequency features while reducing the number of modes. After completing the FNO model training, the test area was similarly divided into discrete sections, ensuring no direct contact between adjacent sections. The FNO was then employed to independently predict the wind field for each section, and the individual predictions were combined to generate a complete wind field representation of the urban area. Experimental results demonstrated that the FNO method based on small-block training was more effective than the method based on larger blocks with fewer modes, improving prediction accuracy by 50% over a 7-second wind field simulation, while also reducing the time required for single-step predictions. Notably, we also incorporated Signed Distance Function (SDF) data, which resulted in approximately a 60% increase in prediction accuracy during the first 7 seconds of output. Few researchers have previously applied neural operators to fluid simulation by dividing blocks, making this work a novel approach to training scientific computations in deep learning under low-performance hardware conditions.

Chapter 2 Literature Review

In recent years, artificial intelligence (AI) technology has not only made a big splash in the field of computer science [1,2,3,4,5], but also become a research hotspot in many cross-disciplinary applications [6,7,8,9,10]. There is an influx of researchers in the fields of computer vision, materials, aerospace, etc. [11,12,13,14], and among them, deep learning techniques are especially suitable for finding implicit data connections in large amounts of data, so deep learning-based methods have become an important means of data connection mining. Deep learning methods have an immeasurable potential for the simulation of micro-city wind s and the CFD field in general [15]. To facilitate the description of our work, this chapter first introduces several recent fundamental methods for micro urban wind field simulation, mainly including FFD, PMM, MM and CityFFD [16], which are all based on models optimizing the solution process of the Navier-Stokes equations; then introduces recent deep learning-based deep learning methods with CFD application cases, including PINN and neural operator methods, and finally clarify why we choose the FNO method and the problem we want to solve.

2.1 Fast Urban Airflow Simulation

For urban wind simulation, due to the design of complex building geometry [17] and complex mesh generation requirements [18], the computational cost of traditional CFD simulation methods is high, so related researchers are looking for methods that can optimize and simplify the traditional CFD simulation methods in order to simulate the urban wind simulation more efficiently, and the following are a few methods that appeared in recent years and received a lot of attention.

2.1.1 FFD

With the development of computer resources, computational fluid dynamics is increasingly used to study airflow and pollutant dispersion around buildings, from individual buildings to city blocks [19], CFD simulation methods provide data for the entire flow field by solving the Navier-Stokes equations, however, the computational efficiency of CFD methods is still difficult to meet the high demand of simulation for industrial applications, for example [20], which was simulated for more than 100 hours. In contrast, Fast Fluid Dynamics (FFD), proposed by Stam in 1999 [21], is a fast simulation method that can provide airflow and contamination dispersion, which was firstly applied to airflow simulation in video games. FFD reduces numerical diffusion by tracking the motion rules of fluid particles in time steps through a semi-Lagrangian method, and avoids the numerical instability that may be brought by explicit solutions by implicitly solving for the diffusion and source terms. Numerical instability that may come with the explicit solution is avoided by solving the diffusion and source terms implicitly to improve the computational speed;

The fundamental equation in FFD is derived from the simplified form of the incompressible Navier-Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

Where:

- $\mathbf{u}$ is the velocity field
- p is the pressure field
- ρ is the fluid density
- ν is the kinematic viscosity of the fluid
- $\mathbf{f}$ represents external forces

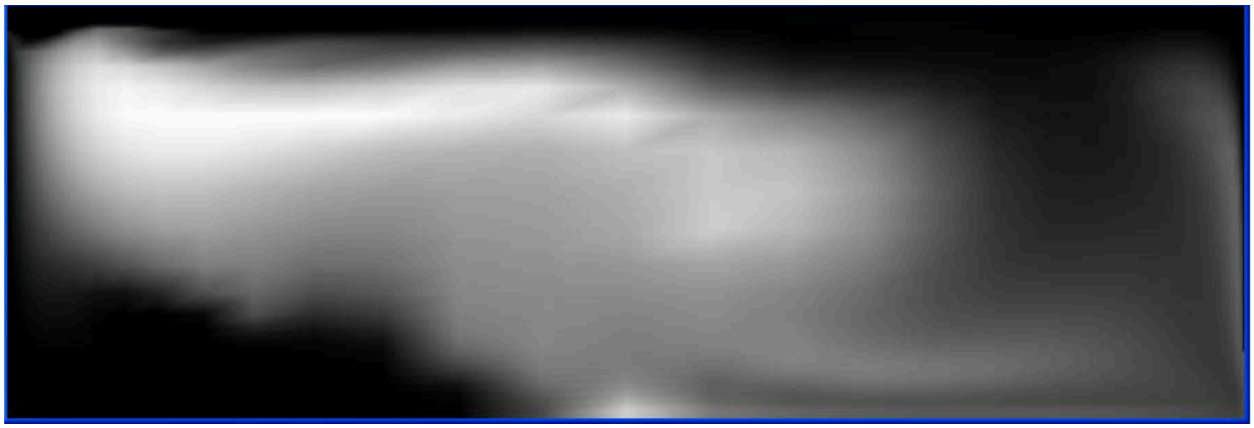

Figure 2.1 FFD simulation [22].

As shown above, in 2001, Zuo et al [22] used FFD to simulate smoke motion in a real industrial application, and they found that FFD is 50 times faster than CFD; because FFD is simpler to implement than CFD and has lower resource requirements, FFD has the potential to be applied to large-scale hydrodynamic simulations; However, due to the simplified numerical scheme, FFD is not as good as CFD methods for tasks requiring high level of detail, and due to the fact that it is based on specific physical assumptions, e.g., low Reynolds number, FFD does not perform well in complex turbulent structures or complex geometries.

2.1.2 PMM

The Porous Media Model (PMM), which considers an entire urban area as a cluster of buildings with different porosities, was originally proposed by Parker [23] et al. Each cluster consists of an array of porous cubes with a single porosity and uniform building height. In the study by Lien [24] et al. the units of the building and fluid portions of the complex are referred to as representative elementary volumes (REVs). There are two main approaches to the definition of REV in PMM applied to urban environments. In the work of Hang [25] et al. two forms of REV were used for micro-turbulence simulation of building complexes and tested the spatially

averaged flow velocity in the horizontal profile, the vertical velocity out of Z=0.5H and the Forchheimer coefficient at Z=H. To better capture the resistance effects within porous media, especially under high Reynolds number conditions, the Forchheimer equation is used to extend Darcy's law. The Forchheimer equation is as follows:

$$\nabla p = \frac{\mu}{K}\mathbf{u} + \beta\rho\,|\mathbf{u}|\,\mathbf{u}$$

where:

- p is the pressure gradient;
- μ is the dynamic viscosity;
- K is the permeability of the porous medium;
- $\mathbf{u}$ is the velocity of the fluid;
- β is the Forchheimer coefficient, representing the strength of the nonlinear resistance;
- ρ is the fluid density.

The Forchheimer equation captures both linear and nonlinear resistance terms, allowing for a more accurate description of flow behaviour within porous media, especially under high flow velocity or turbulent conditions, thereby enhancing the flow simulation capabilities of the PMM model.

In general, the RANS model is the most commonly used model for closing the time-averaged Navier-Stokes equations [26], and the standard k-e turbulence model can be used to evaluate the averaged and turbulent flow characteristics under steady-state, incompressible, and isothermal flow conditions in an urban area; however, in the case of the PMM model, the conventional RANS model does not take into account the surface voids of the porous medium that generate the viscous forces and drag forces, therefore, it is necessary to introduce the reference averaging technique to solve the model specific kernel problem.Hang et al [27] modified the general double equations based on incompressible flow porous media by basing the CFD coefficients in each REV on the calculation of the pressure difference between the surface of the windward wall and the surface of the leeward wall for each building.

The PMM method is computationally less expensive, efficient and requires fewer grids for computation, and can be embedded into already existing CFD tools, which facilitates incumbent CFD practitioners to use PMM directly for urban wind field simulations. However, the accuracy of PMM simulation is limited and depends on the physical characteristics of the urban complex, and the accuracy will decline when the geometrical layout is more complex.

2.1.3 MM

Unlike the PMM model, the MM(Multi-zone Model) model assumes that the urban canopy is a network of interconnected street canyons, and the entire urban agglomeration can be divided into zones, and then the relevant equations for mass, pressure, and energy balances are developed

separately for each zone. In the study of Yao [28] in 2011, five atmospheric zones are newly defined: the urban boundary layer (the part of the city that exceeds 100 meters in the vertical direction) and the lateral boundary layer in four directions. meters) and the lateral boundary layer in four directions.The basic principle of the MM method is to model the heat and air flow equilibrium equations of the urban air flow network in time, and then solve them by numerical methods.

For each zone, the MM method describes mass conservation using the simplified mass balance equation:

$$\sum_{j} (m_{ij} - m_{ji}) = M_i$$

where:

- m_{ij} represents the mass flow rate of air from zone I.
- m_{ji} represents the mass flow rate of air from zone .
- M_i is the mass source or sink in zone i.

The MM method simplifies the modeling of complex urban structures, which is highly efficient in the simulation of large urban areas, especially when the researcher's simulation area needs to simulate the wind-heat environment and energy consumption as a whole, the MM model is able to deal with the simulation of the heat balance and energy consumption, and the UrBEC model using the MM method is on average 5 times faster than the Ansys Fluent and 20 times faster than the ENVI-met platform for the same simulation scenario. ENVI-met platform block by 20 times. However, due to the neglect of vertical differences within the region, this approach may have insufficient accuracy when dealing with complex urban layouts.

2.1.4 CityFFD

Wang [29] et al. developed a new urban climate model, CityFFD, for fast and accurate simulation of urban microclimates, which includes some new higher-order temporal and spatial schemes to achieve accurate results on coarse grids.

The traditional semi-Lagrangian method is based on a linear interpolation scheme, which leads to high dissipation errors [30], to solve this problem, CityFFD uses a higher-order interpolation scheme based on two third-order polynomials to reduce the high dissipation errors; CityFFD includes some new higher-order temporal and spatial schemes to achieve accurate results on coarse grids [31,32,33,34]. ; in addition to this, CityFFD takes advantage of multi-core CPU (i.e. OpenMP) and GPU computing to alleviate the high dependence on supercomputers typically required for previous urban microclimate simulations. It makes it possible for researchers to model entire cities using personal desktop computers [35,36]. By saving computational time and resources, CityFFD also opens up new possibilities for capturing complex turbulence phenomena in urban microclimates using LES [37]. Fast and accurate models like CityFFD that run on personal computers make it relatively easy to develop integrated multi-scale simulation tools (e.g. mesoscale and building models) [38]. It helps to

integrate atmospheric and climate models with building models to meet the demand for advanced multiscale modeling of urban systems.

CityFFD solves the following three conservation equations, where $\vec{U}$, θ, and p, Re, Gr, Pr, ν_t, and α_t represent the velocity, temperature, pressure, Reynolds number, Grashof number, Prandtl number, turbulent viscosity, and turbulent thermal diffusivity, respectively:

$$\nabla \cdot \overrightarrow{U} = 0$$

$$\frac{\partial \overrightarrow{U}}{\partial t} + \left(\overrightarrow{U} \cdot \nabla\right) \overrightarrow{U} = -\nabla p + \left(\frac{1}{Re} + \nu_t\right) \nabla^2 \overrightarrow{U} - \frac{Gr}{Re^2}\theta$$

$$\frac{\partial \theta}{\partial t} + \left(\overrightarrow{U} \cdot \nabla\right) \theta = \left(\frac{1}{Re \cdot Pr} + \alpha_t\right) \nabla^2 \theta$$

The detection terms in equation 1 are addressed using the Lagrangian method, where values for the unknown variables (e.g. velocity and temperature) at position s_c^{n+1} are determined by calculating the values of $\vec{U}$ and θ at position S_c^n, as illustrated in equation below:

$$\mathrm{S}_c = \overrightarrow{U} dt \rightarrow S_c^n \approx S_c^{n+1} - \overrightarrow{U} \Delta t$$

To capture the turbulence behaviour, a large eddy simulation (LES) turbulence model is applied, and the turbulent viscosity is calculated using equation below.

$$\nu_t = \left(c_s \Delta\right)^2 |\mathbf{S}|$$

In this equation, c_s, Δ, and S represent the Smagorinsky constant, filter scale, and large-scale strain rate, respectively, with c_s typically ranging from 0.1 to 0.24. CityFFD is equipped with a 4th-order interpolation scheme to model airflow on coarse grids and mitigate high dissipation errors. Comprehensive details of the method are available in earlier literature [29].

2.2 AI Technologies

Since the 21st century, with the development of GPU hardware, the theory and technology related to AI have gained a great deal of momentum, and deep learning has made a big splash in various fields due to its powerful ability to search for implicit associations in data and its low need for explicit mathematical formulas. Common deep learning models are CNN, GNN, RNN and Transformer [39,40,41,42]. In addition, deep learning has made breakthroughs in combining with reinforcement learning, and graph neural networks, resulting in research areas such as deep reinforcement learning [43]. The rapid development of deep learning provides new ideas and methods for miniature urban wind simulation, and the core idea of using deep learning methods to improve the efficiency of CFD simulation of urban wind s is based on the neural network to find and construct the mapping function from input wind data to output wind field data, thus

accelerating the iterative computation process of the CFD solver, which is different from the traditional CFD simulation methods, and the deep neural network does not need to explicitly provide the parameters of the equations that need to be found, but only need to give a large amount of training data, and the deep neural network will learn the representation from a given input to the corresponding output in the large amount of training data.

For the characteristics of the flow field data, we found that employing the image-to-image [44] method in computer vision is particularly suitable for our wind field simulation task. In this section, we will focus on two mainstream AI for Science methods and explain why we chose the FNO neural operator as our method.

2.2.1 Physics-Informed Neural Network (PINN)

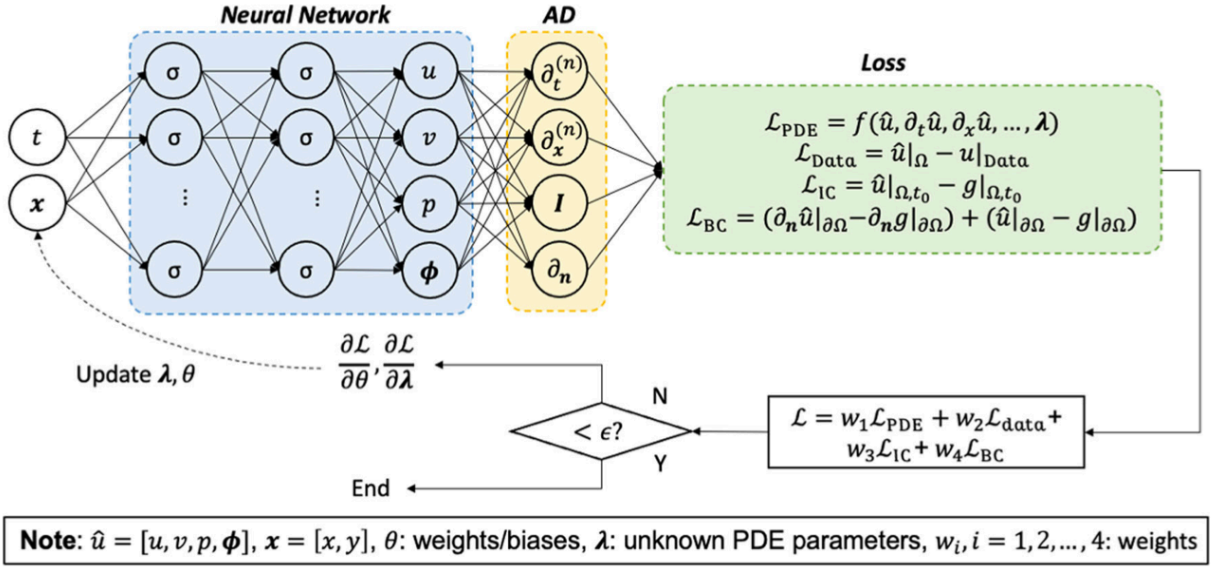

Figure 2.2 PINN architecture [67].

Physics-Informed Neural Network (PINN) is a machine learning technique used in scientific computing for solving problems involving Partial Differential Equations (PDEs). PINN approximates the solution of PDEs by training a neural network to minimize a loss function.Unlike traditional ANNs, it incorporates a physical loss term of the desired PDE in the loss function, which includes initial conditions reflecting the convenience along the spatiotemporal domain and the boundary conditions as well as terms that reflect the residuals of the partial differential equation at selected points in the domain. The trained PINN can generate the estimated solution at a given input point in the domain of function definition, since the solution is determined directly from the physical equations, it can be considered as an unsupervised strategy. Originally PINN was proposed by the work of Raissi [45], who devised purely physics-driven methods for inferring solutions to general nonlinear partial differential equations and constructing computationally efficient models of physical agents, which not only

demonstrates a range of promising approaches in scientific computing, but also opens up a path between the worlds of deep learning and mathematical physics modeling. This is in line with the trend of deep learning techniques in recent years, and is a timely contribution that can benefit practitioners in a wide range of scientific fields to easily enjoy these benefits for applications including, but not limited to, model predictive control, multi-physics modeling, and simulation.

However, PINN still has some pending challenges. Numerical methods such as finite element, spectral methods, etc. have matured over the past 50 years, and in many cases they fulfill the robustness and computational efficiency criteria required by industrial practice [46,47]. Neural networks are limited by the lack of mathematical logic and frequency bias in the network itself, and are not comparable in accuracy and stability to PDE solvers that are backed by explicit mathematical logic. This leads to the fact that even extremely small deviations from the initial conditions may result in extremely large deviations from the PINN, and secondly, the ability of the PINN to handle problems with inhomogeneous parameter space is still extremely limited, e.g. for the problem of NS equations, fewer PINN methods are available to handle high Reynolds numbers. Learning effectiveness for the PDE parameter inhomogeneity case is also a major challenge for PINN. As a result, purely data-driven neural operator methods have emerged, where neural operator learning aims at exploring the properties of the underlying dynamical system or partial differential equations from the given simulation or real PDE data [48], and we discuss the definition of neural operators and the main neural operator methods in recent years in the next chapter.

2.2.2 DeepONet

Hornik et al. have shown that neural networks with a single hidden layer can accurately approximate any nonlinear continuous operator [49]. However, this theorem only guarantees the approximation error under a sufficiently large network, and does not take into account the generalization error and computational cost during the actual optimization process. To address these issues, Lu et al. [50] proposed Deep Operator Networks (DeepONets), which aim to learn a broader range of continuous nonlinear operators, improving the performance of neural networks in handling nonlinear problems.

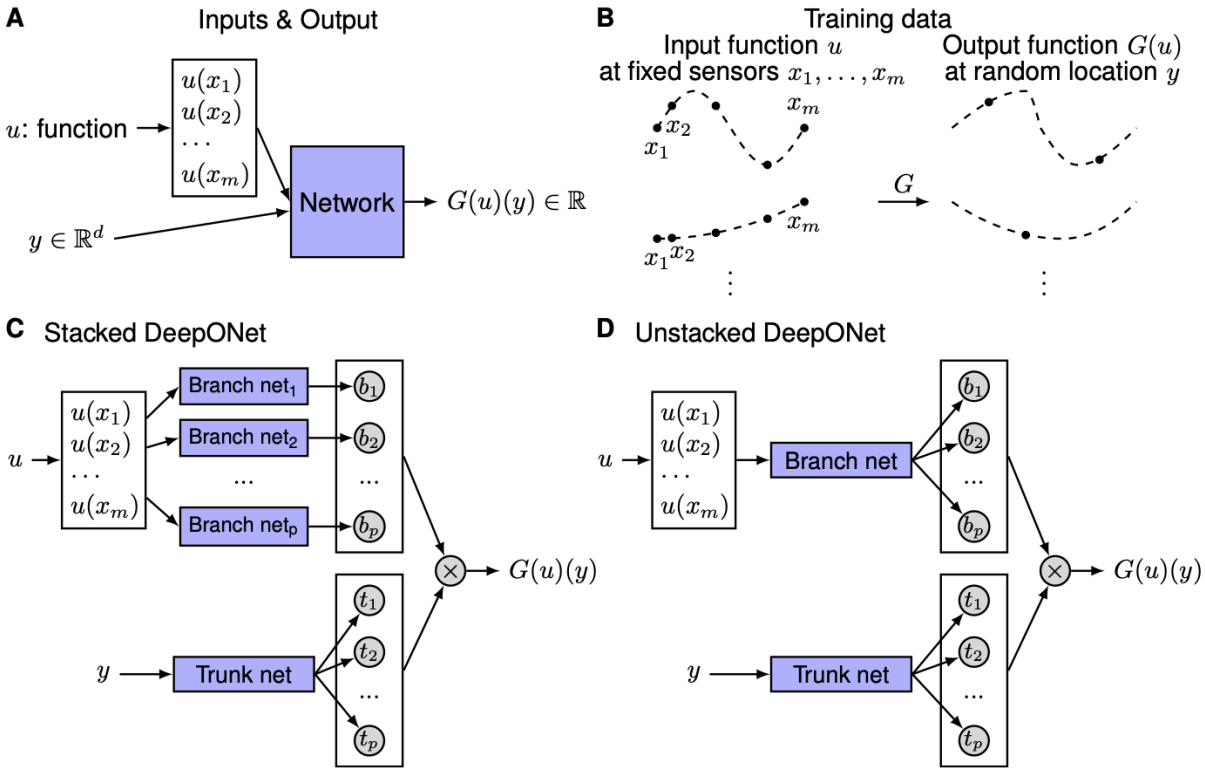

Figure 2.3 Different DeepONet architectures [50].

The above figure shows the model structure of DeepONet, A above is the most basic DeepONet structure, C and D denote the Stacked and Unstacked DeepONet respectively; are the corresponding output functions. For any point in the domain, the output is a real number. In practice, we represent these input functions discretely in order to apply network approximations. Here we explore different ways of representing functions in the input space. The simplest one is based on a sufficient but finite number of function values at locations x_1,x_2...x_n called "sensors" in the structure of the DeepONet, as shown in Figure B above. There are other ways to represent a function, such as using spectral expansion or as an image, DeepONet is divided into an offline training phase and an online inference phase, in the offline phase DeepONet uses data computed using classical numerical methods as training data, depending on the size of the data, DeepONet requires from one to hundreds of Graphics Processing Units (GPUs) hours, and it can use experimental or simulation data for training at different scales and accuracies. And DeepONet trained in the online phase can be used as an alternative to CFD methods because the process does not require re-training and only involves forward passes of the network, and thus can be used for high-dimensional data to accelerate computationally intensive applications.

$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), \quad x \in [0,1], t \in [0,1]$$

The above equation is the diffusion-reaction partial differential equation where D=0.01 is the diffusion coefficient and k=0.01 is the reaction rate. DeepONets learn the operator from (x) to the PDE solving $s(x, t)$, in this case multiple random points are used for training.To generate the training dataset the second order implicit finite difference method is used to solve the diffusion-reaction system with a grid size of 100x100 and then P randomly selected points in this grid range are used as training data. When DeepONet is trained with only 100 u-samples, the test error can reach the 10-5 level, which demonstrates the potential of DeepONet to be used for small-sample learning [51].

2.2.3 Convolutional Neural Operator (CNO)

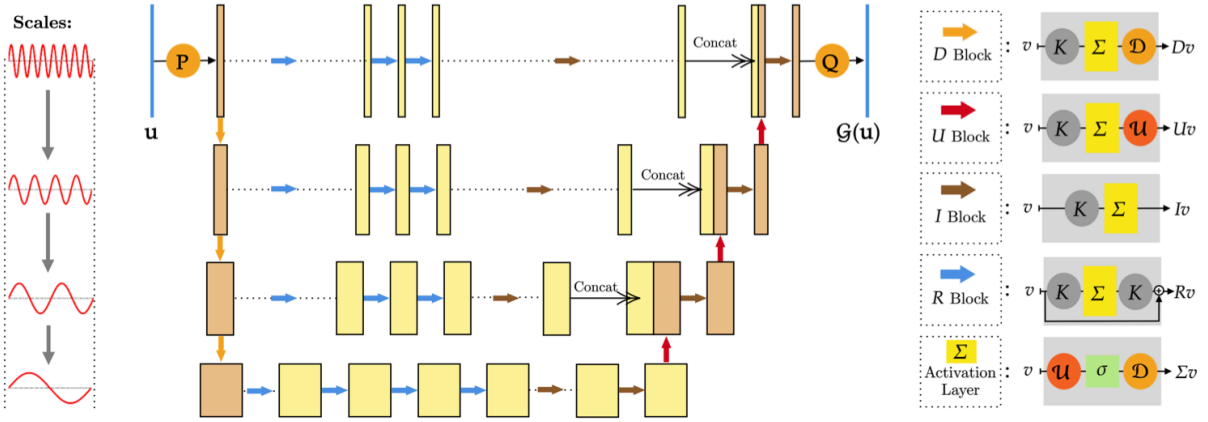

Figure 2.4 CNO architecture [52].

The CNO model architecture is shown in the figure above, CNO is a neural operator based on deep learning method proposed by [52] et al. Unlike FNO which uses Fourier module [53] to extract the frequency features, CNOs are directly convolved in the physical space which makes the operation more intuitive and easier to adapt to a variety of different types of data, before inputting to the activation function, the CNOs performs the up and down sampling and the feature merging process to combine features of different dimensions, which facilitates the retention of features of different dimensions, this structure is inspired by the U-Net structure proposed by Ronneberger et al. U-Net was initially used in the field of medical image segmentation, and is now being borrowed by neural networks in other fields [54]. There are four Blocks in the above figure: Downsampling (D) Block downsampling block; Upsampling (U) Block upsampling block; ResNet (R) Block residual block and Invariant (I) Block invariant block, u is the input function, is a vector, which is first dimensionally increased by a linear layer The number of channels is increased by a linear layer, and then downsampling is used to reduce the resolution while extracting higher dimensional features, which are finally merged into a

single vector after the residual module and upsampling.

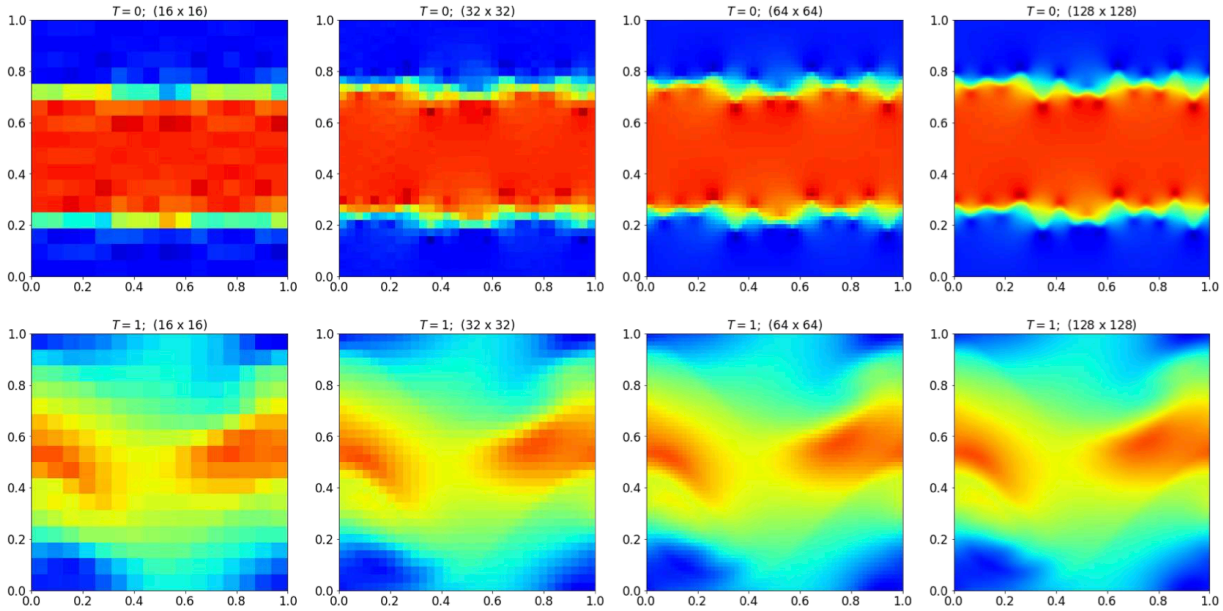

Figure 2.5 An example of input and output (ground truth) samples at 4 different resolutions for the NS experiment [37].

From the Figure 2.5, it can be seen that CNO also has excellent generalization ability, with low prediction error on different resolution data in the test of solving NS equation. Raonic et al. also did a comparison between CNO and other NO methods in terms of performance, computational efficiency, out-of-distribution generalization, and data scaling, etc., and the comparison results show that CNO is basically at the optimum, which provides a possibility for its future application in science and engineering fields. This provides a possibility for its future application in science and engineering, however, CNO still requires high computational resources, and the computational cost will increase exponentially when scaling to 3D space, and is affected by the curse of dimensionality [55], so this will be the direction of CNO's further research in the future.

2.2.4 FourCastNet (FCN)

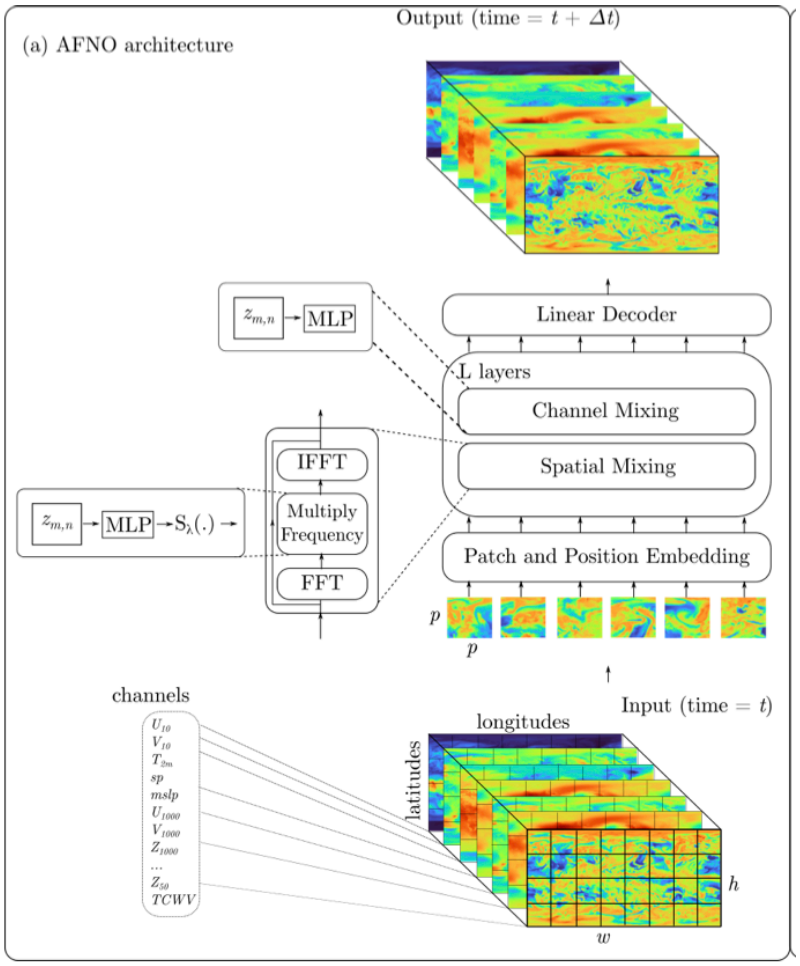

Figure 2.6 FourCastNet structure [56].

FourCastNet is a neural operator model based on Adaptive Fourier Neural Operators (AFNO), proposed by Pathak et al. [56,57]. In recent years, the Vision Transformer (ViT) architecture [60] and its variants have made significant advancements in computer vision, performing well in tasks such as image recognition, image segmentation, and video prediction [58,59,60]. However, the ViT model requires quadratic token operations for high-resolution images, leading to exponentially increased memory consumption as the resolution grows, making it difficult to apply directly to high-resolution meteorological data prediction. In contrast, AFNO's strength lies in transforming token mixing operations into global convolutions in the Fourier domain, efficiently implemented via Fast Fourier Transform (FFT). This gives AFNO greater flexibility and scalability in both spatial and channel dimensions, while reducing the time complexity of spatial mixing to O($NlogN$), where N is the number of tokens, significantly lowering the computational cost.

The model structure and workflow is shown in the figure above, where the input data is

first projected onto a two-dimensional network ($h * w$) of small patches, each represented as a d-dimensional token, and then the sequence consisting of cuts into small blocks is sent together to a series of AFNO layers, each given an input tensor X ($H * w * d$) of blocks for spatial mixing and channel mixing. Finally, an inverse Fourier transform is performed to obtain the output. The experimental results show that the prediction speed of FourCastNet is four to five orders of magnitude faster than the traditional numerical weather prediction (NWP) model, which suggests that FourCastNet has the prospect of replacing or assisting the NWP in the industrial land, however, due to the lack of data assimilation setup, FourCastNet is not yet able to perform real-time weather prediction, and the training FourCastNet is a purely data-driven deep learning model, which requires a lot of computational resources, and there are still deficiencies in the training phase and interpretability, which will be a challenge for FourCastNet to face in the next stage.

2.2.5 Fourier Neural Operator (FNO)

Neural operators do not need to know the details of the underlying partial differential equations, and only need to find the implicit PDE relationships in the data, but previous neural operators are generally less computationally efficient than CNN/RNN [61]. The Fourier transform is commonly used in spectral methods for solving differential equations because differentiation is equivalent to multiplication in the Fourier domain, and the Fourier transform has played an important role in the development of deep learning, and there has been work on using the Fourier transform for accelerating convolutional neural networks such as neural network architectures involving the use of the Fourier transform, and the FNO, a neural network architecture based on the Fourier transform, which was proposed by Li [62] et al. FNO is a new deep learning architecture that is capable of learning mappings between functions in an infinite dimensional space.

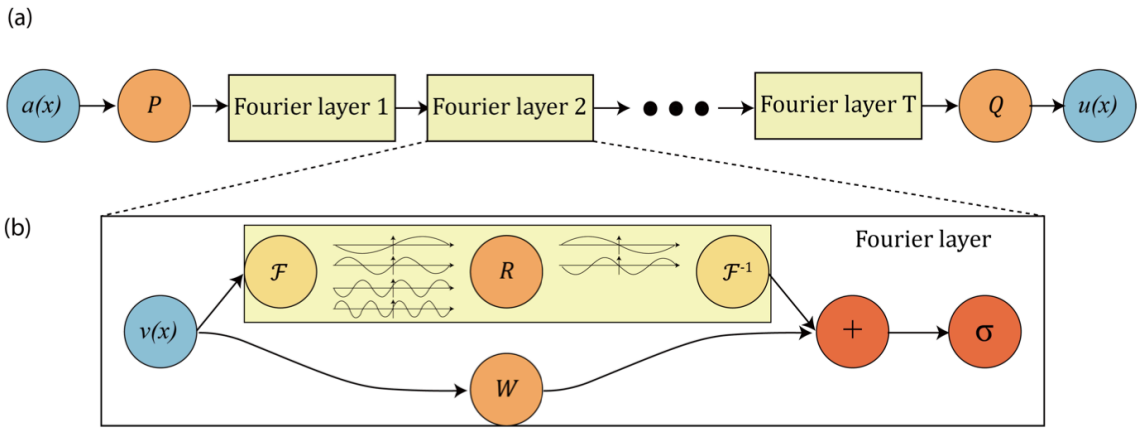

Figure 2.7 FNO architecture [62].

The model architecture of FNO is shown in the figure above, starting from the input a(x), which is boosted to the channel space of higher dimensions through the linear layer P, then four Fourier layers and activation functions are applied, and finally passed back to the target dimensions through the linear layer Q. The Fourier layer consists of upper and lower circuits; in the upper circuit, the input undergoes a single Fourier transform F, and then a linear transform R is applied to the low-frequency signals, which filters out the signals of higher frequencies and then the Fourier inverse transform F^{-1} is applied. In the lower circuit a local linear transform W is applied and finally the results of the upper and lower circuits are summed to the output. The Fourier layer is discretization invariant since it can learn and evaluate functions that are discretized in an arbitrary way. Since the parameters are learned directly in Fourier space, analyzing the function in physical space requires only the projection, and these bases are well-defined everywhere on the top, so the FNO is capable of zero-point super-resolution; after learning on simulated fluid data with a fixed resolution of 64 * 64, the FNO can be reasoned about in 0.005 seconds, whereas the pseudo-spectral method takes 2.2 seconds to solve the NS equations, which shows that the FNO has a huge speed advantage, and FNO learned on 64 * 64 resolution data can be inferred on 256 * 256 data without any degradation in accuracy, which indicates that FNO has the ability to learn resolution invariant solutions for the family of NS equations in turbulent flow.

Compared with PINN, the FNO method only requires pure data without physical error and has better generalization ability, and since our data amount is not large enough and GPU resources are limited, FNO performs better than FourCastNet, while the training efficiency and inference time under the same data volume are better than DeepONet and CNO, therefore, we choose FNO as our experimental method

2.3 Summary

In this chapter, we first clarify the problems in CFD for micro-urban wind field simulation to be solved by deep learning techniques and methods by comparing the workflow of traditional CFD solvers and deep learning-based micro-urban wind field simulation, and then introduce several mainstream CFD methods for urban wind field simulation, with a focus on how they describe the fluid motion laws at the macroscopic and microscopic levels. Finally, the main AI which can be used for CFD simulation in recent years are introduced, focusing on the Neural Operator method and the reason why we chose the FNO method.

Chapter 3 Methodology

This chapter introduces the Fourier neural operator FNO method based on deep learning and how to apply it to urban wind field prediction tasks. First, the urban wind field data generated by CityFFD is expressed as an input acceptable to the deep neural network through appropriate methods, and then based on Depending on the GPU hardware resources and training efficiency requirements, we generate a data set by cutting the training data into small patches, and splice the predicted flow field of the test set into a complete predicted flow field in the urban area. We predict the flow field in different wind directions and different urban geometries. A comprehensive generalization prediction test was conducted on different wind directions and different urban pattern, and an ablation study was conducted to explore the role of the Fourier module.

3.1 Overview of Methodology

In 2020, Zong Yili [62] team adopted the FNO network. This kind of network is different from traditional solvers that solve equations through discrete spaces, such as the finite element method (FEM) [46] and the finite difference method (FDM) [63]. Traditional methods require a trade-off in resolution: coarse meshes are fast but have low accuracy, and fine meshes have high accuracy but are slow. FNO networks overcome the reliance of traditional numerical methods on grids by generating a set of grid parameters that can be used for different discretization. It can be solved at different grid sizes, and FNO only needs to be trained once to predict new inputs. Prior to FNO, no purely data-driven method could compete with CFD methods in limited-dimensional settings. FNO solves this problem by extracting feature components through fast Fourier transform.

3.2 Training Data Preparation

The first problem to be solved when using the FNO method based on deep learning technology to predict micro-city wind fields is the representation of flow field data, that is, how to represent boundary conditions, physical fields (such as velocity vector fields), etc. into something acceptable to the neural network. form. Here we introduce the distance function (Signed Distance Function, SDF) and binary representation (binary representation)

Signed distance function For a two-dimensional Cartesian image, each Cartesian grid point on the image domain $\Omega \in$ R2 is $(i : j)$, and use $f(i : j)$ to represent the signed distance function sign. For points through which the geometry passes, the value of f (i; j) is 0. Suppose the set of boundary points is Z, then:

$$Z = \{(i, j) \in \mathbb{R}^2 : f(i, j) = 0\}$$

When the point is inside the geometry, $f(i; j) < 0$, when When the point is outside the geometry, $f(i; j)) > 0$, the specific calculation formula of the signed distance function $D(i : j)$ is as follows:

$$D(i,j) = \min_{(i',j')\in Z} \left|(i,j) - (i',j')\right| \operatorname{sign}(f(i,j))$$

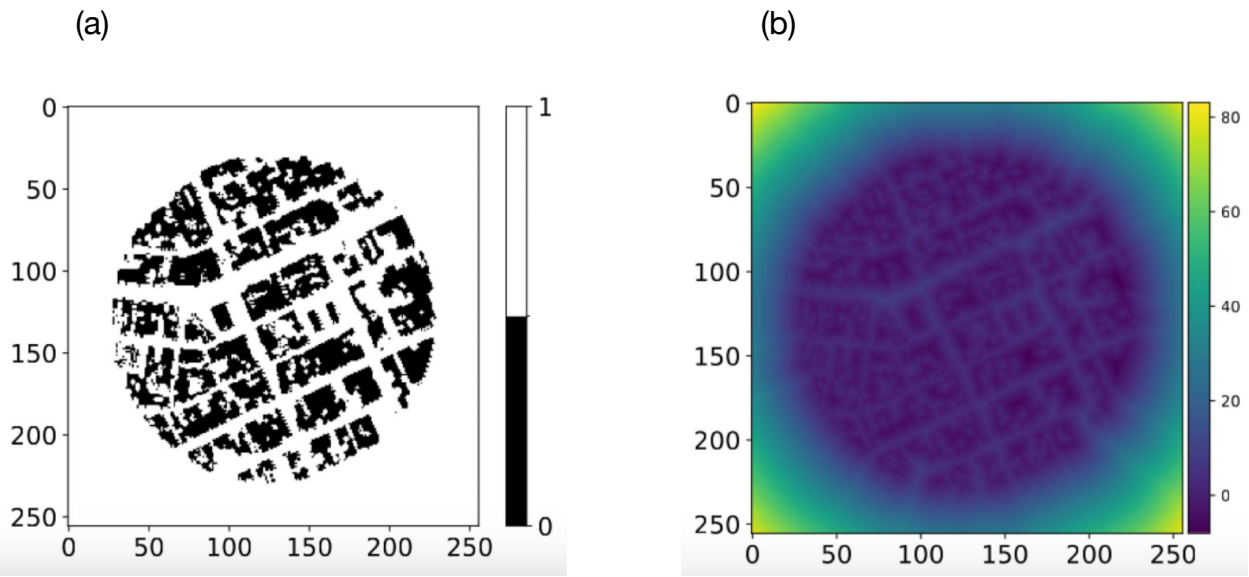

Figure 3.1 (a).binary image. (b).SDF of Niigata.

$D(i:j)$ represents the shortest distance from a given point $(i:j)$ to the geometric boundary. Figure 3.1 represent the binary image and SDF of Niigata. First, the geometric figure is projected onto the Cartesian grid, and $B(i:j)$ is used to represent the binary method. Then when the point is inside or on the boundary of the geometry, $B(i:j) = 1$. When the point is outside the geometry, $B(i:j) = 0$. This representation can represent the geometry as an "artificial image". Compared with SDF, the binary method is further simplified, and the physical distance from each point to the boundary is not explicitly represented.

For the west wind field data of Niigata City, we performed the following preprocessing. The original data has a resolution of 256 * 256 and contains a total of 1200 time steps. Since the wind has not yet covered the entire urban area within the first 180 steps, we discarded the wind field simulation data within the first 180 time steps in order to ensure the learning effect of the model. Each time step is 0.1 seconds.

We divide the wind field data into 64 * 64 grid patches. On this basis, 80 positions were randomly selected for training. For each location, the model uses 5 time steps of data as input and predicts 10 time steps of data as output. Specifically, for a given location, we input the data from steps 181 to 185 and output the data from steps 186 to 195. Then, we move two steps at a time, input the data from steps 183 to 187, and output the data from steps 188 to 197, and so on until step 1200.

To ensure comprehensive coverage of data, we conducted multiple sampling processes to ensure that 80 locations covered more than 90% of Niigata urban area. The above process constitutes our data processing process, and we hope to provide sufficient and effective training data for the deep learning model.

3.3 FNO Model Architecture

3.3.1 CNN Layers

In the FNO benchmark network, the convolutional layer is also an important component [39,62]. A typical convolutional layer is responsible for extracting features. The convolution process regards the convolution kernel as a filter for subsequent work, processes the selected area of the input image data, and then obtains a simplified expression of the characteristics of the corresponding area. From a mathematical point of view, convolution is to multiply the matrix of the input image after processing and the matrix of the convolution kernel. A convolution operation outputs a corresponding value, and the entire convolution process is to convert the selected convolution kernel The matrix is slid on the processed matrix of the input image, and finally a new matrix is obtained, as shown in the figure below [37] :

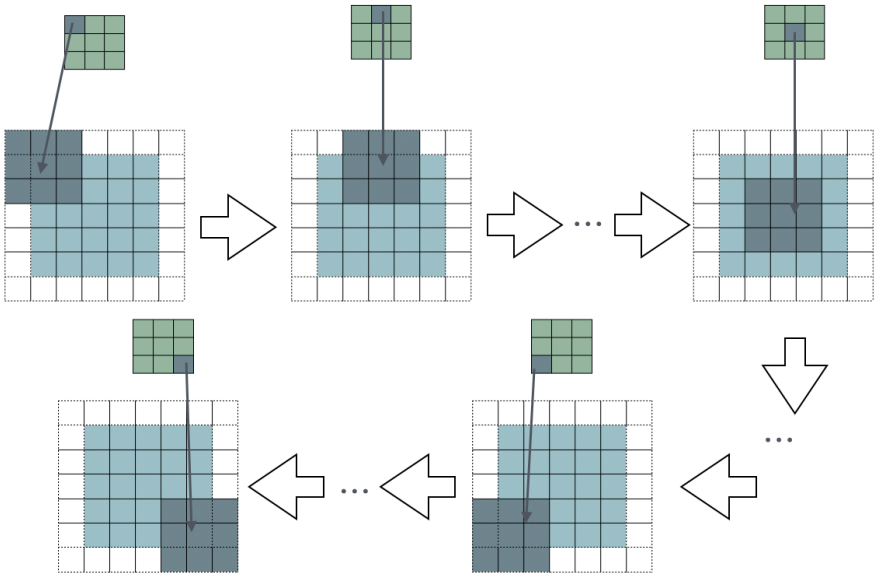

Figure 3.2 Convolutional kernel [39].

Common convolution processes generally include the following four parameters:

① The size of the convolution kernel: The size of the convolution kernel represents the range of its receptive field in the neural network There are many types of convolution kernels. The setting of the convolution kernel should be set according to the specific network structure size to complete the extraction task. As shown in the figure above, the convolution kernel in the convolution operation process diagram is 3 * 3.

② Step size: The step size represents the accuracy during the convolution process, which represents the distance spanned by each movement of the convolution kernel on the matrix after processing of the input image. Under normal circumstances, the step size defaults to 1. As shown

in Figure 3.2, if it is the default step size, the problem of secondary coverage will occur during the movement of the convolution kernel; if the step size is 3, the movement process Then complete coverage cannot be achieved, so the convolution operation in the figure adopts a step size of 2.

③ Padding: During the convolution process, sometimes the convolution kernel does not match the matrix parameters. For example, as shown in Figure 3.2, the matrix size after wind field image processing is 5 * 5, and the kernel is 3 * 3. According to the default step size Precision, the convolution kernel will slide out of the matrix, so the matrix needs to be expanded to 6x6, that is, filling processing.

④ The number of input and output channels: The number of input channels of the convolution kernel is equal to the number of channels of the input matrix, and the number of output channels is equal to the number of output channels of the convolution kernel. Multi-channel sharing reduces the number of parameters to a large extent, and the output feature map is only related to the local input feature. It is not only is retained the powerful feature extraction ability of CNN, but also comprehensively reduce the amount of parameters and the amount of calculation.

3.3.2 Fourier Layers

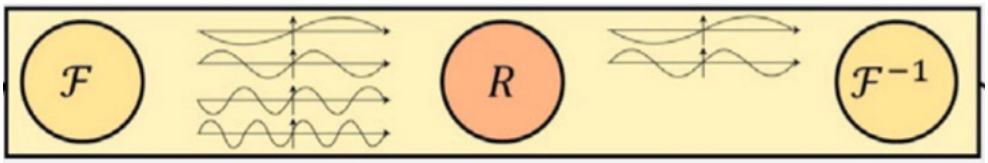

Figure 3.3 Fourier layer [62].

The basic idea of the Fourier layer originates from Fourier analysis, that is, any periodic function can be expressed as a linear combination of sine and cosine functions of different frequencies. In FNO, the Fourier layer as shown in Figure 3.3 converts the input data from the physical domain to the frequency domain by performing a Fast Fourier Transform (FFT) on it. This step reveals the frequency domain characteristics of the input data, providing a richer and deeper representation for subsequent processing. Specifically, the workflow of the Fourier layer can be divided into the following steps:

Frequency domain transformation: First, the input feature data is converted from a physical space representation to a frequency domain space representation using Fast Fourier Transform (FFT). This transformation demonstrates the composition of the input data at different frequencies, making it possible for Fourier layers to capture complex spatial patterns.

Frequency domain operation: In frequency domain space, the transformed features are applied with linear transformation. The Fourier layer multiplies them with a learnable frequency domain weight matrix and filters the frequency components outside the specified number of modes, thereby retaining the desired frequency features.

Inverse Transform: Finally, the transformed frequency domain data is converted back to physical space using the inverse fast Fourier transform (IFFT), and the input data is re-converted into a physical space table.

The introduction of the Fourier layer brings significant advantages to solving PDE: the Fourier layer can learn the global characteristic frequency of the input data in the frequency domain, and is not limited to the physical coverage of the convolution kernel of the CNN, thus improving the model's learning ability.

3.3.3 ResNet Connection

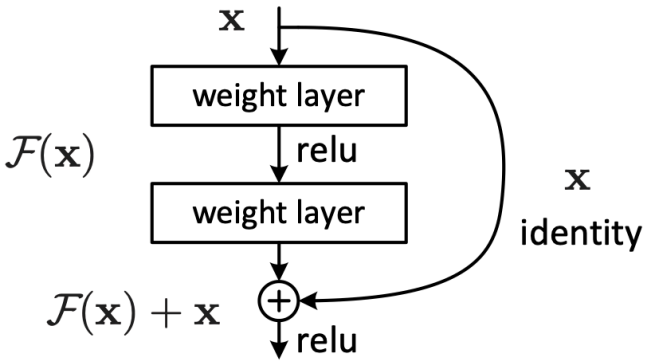

Figure 3.4 ResNet layer architecture[65].

The ResNet network was originally proposed by a research team headed by He Kaiming of Microsoft Research [65]. It showed its talents in the computer vision competition in 2015 and won the championship. The ResNet paper was also selected as the best paper of the CVPR 2016 conference. Excellent paper. In the early days of deep learning research, many researchers found that when the number of network layers reached a certain level, gradients would disappear or explode [39]. This constrained the number of network layers in deep learning, and the emergence of ResNet solved the problem. Solving this problem, it makes effective training of deep networks possible.

The core advantage of the ReNet network is its ability to have identity mapping inside the neural network. As shown in the figure above, this is the core part of ResNet. This module solves the negative problems caused by the increase in network depth in reality: network degradation. The ResNet module changes the model training process from $H(x)=F(x)$ to $H(x)=F(x)+x$, In this way, when $F(x)=0$, the function process will become the identity mapping $H(x)=x$, which effectively prevents the problem of gradient disappearance or gradient explosion in deep neural networks, and allows the number of network layers to be increased than before.

3.3.4 Network Configuration

Table 3.1 Network Configuration

Layer	Kernel Size	Input Size	Output Size	Kernel Number	Parameters
P1	8	64x64x25	64x64x40	40	360
SpectralConv 0	1x1	40x64x64	40x64x64	40->40	2,163,200
SpectralConv 1	1x1	40x64x64	40x64x64	40->40	2,163,200
SpectralConv 2	1x1	40x64x64	40x64x64	40->40	2,163,200
SpectralConv 3	1x1	40x64x64	40x64x64	40->40	2,163,200
MLP0	1x1	40x64x64	40x64x64	40->40	3280
MLP1	1x1	40x64x64	40x64x64	40->40	3280
MLP2	1x1	40x64x64	40x64x64	40->40	3280
MLP3	1x1	40x64x64	40x64x64	40->40	3280
W0	1x1	40x64x64	40x64x64	40->40	1640
W1	1x1	40x64x64	40x64x64	40->40	1640
W2	1x1	40x64x64	40x64x64	40->40	1640
W3	1x1	40x64x64	40x64x64	40->40	1640
Norm0	——	40x64x64	40x64x64	——	——
Q0	40->160 160->10	40x64x64	10x64x64	2	10585
Total					15,187,785

Table 3.1 above shows the specific details of the FNO network structure used in this article. For input and output, 64 * 64 * 5 and 64 * 64 * 10 respectively represent feature maps with 5 and 10 channels and a size of 64x64. For the P1 layer, 64 * 64 * 40 means increasing the number of input vector channels from 5 to 40. In the four Fourier modules, each input feature will be summed after two passes. The first pass is the SpectralConv layer. In the SpectralConv layer, the input features will first be converted into Fourier space representation, and frequency components other than the modes (32) number will be discarded, and the remaining Fourier inversion is converted into a physical space representation and then linearly transformed through the convolution layer MLP; the second path is the W layer, which performs convolution transformation on the input features to extract features and then adds them to the first output. After four layers of Fourier layers, the number of channels is changed to 10 through the linear layer Q, which represents the predicted flow field data of 10 consecutive time steps.

3.4 Parameter Selection

In this study, we used the Fourier neural operator (FNO) model for urban wind field prediction. The FNO model consists of four Fourier modules, each module includes the addition of two outputs of a Fourier layer and a convolutional neural network (CNN) layer. The following is a detailed description of the parameter selection and tuning process.

1). Model parameters :

• Four Fourier modules: Each module consists of a Fourier layer and a CNN layer, and their outputs are summed through a residual connection (ResNet connection).

• Activation function: Use Gelu activation function after the output of each layer to enhance the nonlinear expression ability of the model.

• Total number of parameters: The entire model contains approximately 3.8 million parameters.

2). Training parameters :

• Learning rate: set to 0.01. The initial learning rate is chosen based on experience and a small range of pre-experiments to ensure that the model converges quickly early in training.

• Optimizer: Adam optimizer selected. The Adam optimizer combines momentum and adaptive learning rate adjustment and is suitable for processing large-scale data and complex models.

• Batch Size: Set to 100. Larger batch sizes can fully utilize the computing power of the GPU and help stabilize gradient updates. But too large batch size may affect the learning effect. After several experiments we have chosen 100 based on the results.

• Number of training epochs : The model was trained for 100 epochs. Based on experimental results, 100 epochs of training can ensure convergence while avoiding overfitting.

3). Dataset information :

• Training data set: The data set size is 39600 samples, and the size of each sample is 64x64x15.

• Data preprocessing: Cut the original 256x256 flow field into 64x64 small blocks and calculate the Signature Distance Function (SDF) data. The training set contains 39600 small blocks, the first 5 time steps are used as input, and the next 10 time steps are predicted.

• Data augmentation: Due to the complexity of urban wind field data, we perform data augmentation through patching and SDF calculations to ensure the diversity and representativeness of the training set.

4). Hyperparameter tuning

• Methods: Hyperparameter adjustment is mainly performed through empirical adjustment. Different learning rate and batch size combinations were tested in preliminary experiments, and the current parameter configuration was finally chosen.

• Result evaluation: Choose the best parameter combination by evaluating the performance (MAE) on the validation set.

5). Hardware and software environment

• Computing resources: NVIDIA A100 GPU (32GB VRAM)was used for training. The A100 GPU provides powerful computing power that can significantly accelerate the training process of large-scale models.

• Training platform: The PyTorch framework is used for model construction and training. PyTorch has a flexible dynamic calculation graph and rich community resources, suitable for developing and experimenting with complex neural networks.

• Training time: A single training run of 100 rounds takes approximately 1.5 hours.

3.5 Training and Testing Process

In this study, in order to speed up data access and improve training efficiency, the training data is first loaded into memory. The training process uses an NVIDIA A100 GPU with 32GB RAM. This high-performance computing resource ensures that the model can efficiently handle large-scale data and complex calculations, and a single round of training takes about 1.5 hours. Our training process is as follows:

(1). Data loading and preprocessing:

The size of the training dataset is 39,600 samples, and each sample has dimensions of 64 * 64 * 15. Data preprocessing consists of cutting the original 256 * 256 flow field into 64 * 64 tiles and calculating the Signature Distance Function (SDF) data. The data set is divided into an input part and an output part. The input is the data of the first 5 time steps, and the output is the prediction results of the next 10 time steps.

(2) Training process:

We use a batch size of 100 for training, and the FNO model performs feature extraction by passing the input data to the Fourier layer. During the training process, we use the relative error between the predicted 10-step results and the true value as a measure of the loss function to evaluate the accuracy of the model prediction. At the same time, the FNO model uses the back propagation algorithm to calculate the relative error based on the relative error. The error calculates the gradient and updates the model parameters to achieve the purpose of learning the implicit relationship between data. During the training process, use the tqm library to display

errors in training in real time to monitor and evaluate model performance. The tqm library provides detailed training error curves to help adjust training strategies in a timely manner.

(1). Data preparation:

The test data set we use consists of 16 non-overlapping 64 * 64 * 5 wind field data. We splice the 64 * 64 wind field data together from left to right and top to bottom to form the entire Niigata area. This method can achieve wind field prediction in the entire Niigata 256 * 256 urban area.

(2). Prediction and evaluation:

We use the trained FNO model to predict the test data and predict the wind field 150 time steps after the input. We use the mean absolute error (MAE) to evaluate the accuracy of the model, which reflects the difference between the predicted value and the true value. During the test, we used the same hardware resources (NVIDIA A100 GPU) to ensure the fairness of the experimental comparison.

Through the above training and testing process, we ensure that the FNO model has good performance and efficient computing power in complex urban wind field prediction tasks. These processes describe in detail data loading, model training, error calculation, performance monitoring and testing methods in practical applications, providing an important reference for subsequent research and applications.

Table 3.2 Hyper Parameters

Learning Rate	0.001
Optimizer	Adam
Loss Function	Relative Error
Activation Function	Gelu + Relu
Layers	4
Batch Size	100
Learning Rate Decay	50

3.6 Evaluation Metric

This article uses [64] CFD prediction accuracy to compare and evaluate FNO's prediction results for urban wind fields.

(1) Fluent (2) OpenFoam (3) Aysns

The average error of the above CFD method is 0.5 meters per second.

For the prediction results of the FNO model, this article uses the Mean Absolute Error (MAE) to compare the difference between the predicted wind field and the true value, and conduct an overall assessment of the prediction accuracy of the wind field. Our task is more concerned with specific areas. The prediction results, such as the wind field in the urban building

area, are more helpful for the design of the aircraft. Therefore, for the wind field data used in this article, we performed regional segmentation and only included the wind field in the urban area, removing our Don't care about the part of the city outside the wind field where there are no buildings.

Absolute error calculation formula :

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

SDF calculation formula :

$$\mathrm{SDF}(p) = \mathrm{sign}(p) \cdot \min_{q \in S} \|p - q\|$$

3.7 Spectral Analysis

we provide the process to compute the radial energy spectrums. To maintain the total energy, the radial energy spectrum employed in this work differs from the commonly used radial averaged energy spectrum.

- **Compute the 2D Fourier transform**. Use the Fast Fourier Transform (FFT) to transform the input 2D matrix from the spatial domain to the frequency domain.
- **Compute the wave numbers**. For each entry in the Fourier-transformed matrix, compute its wave number, which is the distance of that entry from the center of the Fourier-transformed matrix.
- **Bin the wave numbers.** Each bin represents a range of wave numbers. For the 64 * 64 matrix, we set 32 bins, each with a range of 1.
- **Compute energy in each bin**. For each bin, sum the squared magnitudes of the Fourier coefficients that fall within the corresponding wave number range. This sum represents the energy in that frequency range. The radial energy spectrum redistributes the energy of the 2D spectrum radially, resulting in a clearer visualization.

Chapter 4 Results And Discussion

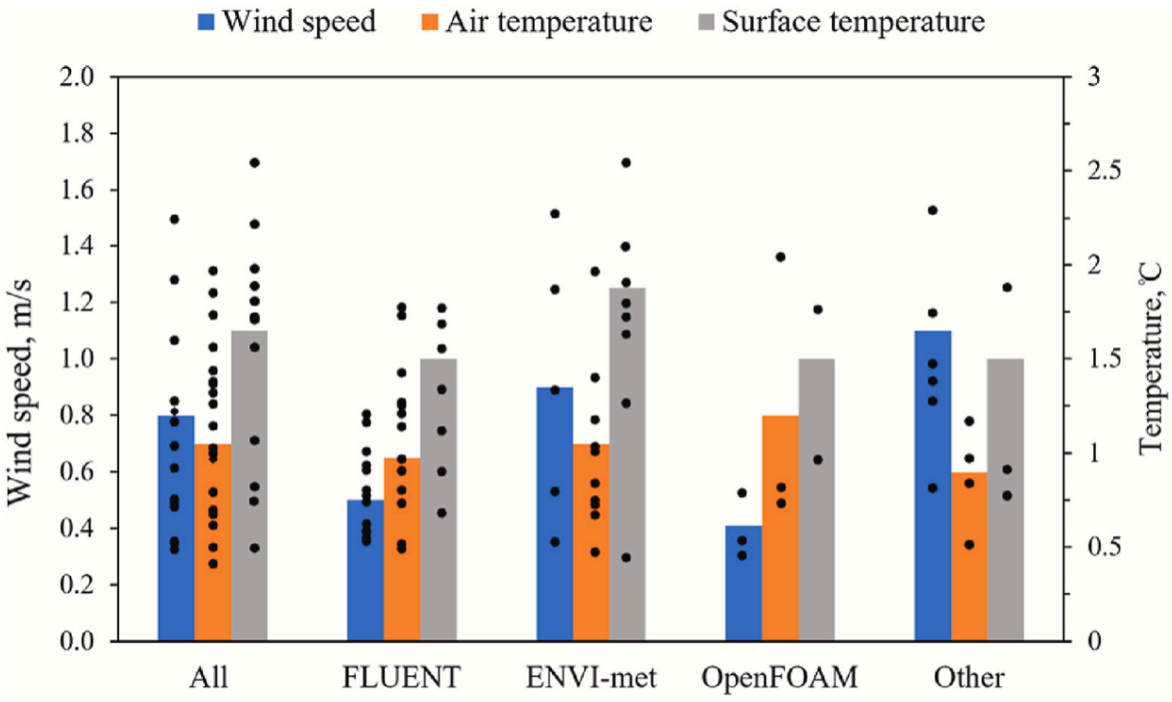

Figure 4.1 Simulation error of wind speed (left axis) and temperature (right axis) by different tools [46].

In this section, we benchmark the performance of our Fourier neural operator (FNO) method against the errors of mainstream computational fluid dynamics (CFD) software in the simulation of urban wind fields. By establishing this benchmark, we can clearly understand the limitations and room for improvement of existing methods in wind field prediction. Figure 1 shows the errors of various CFD software in predicting wind speed, air temperature, and surface temperature. As can be seen, the average error in wind speed (blue bars) is about 0.5 m/s for all software. This benchmark is crucial as it represents the current state of the art regarding the accuracy of CFD methods in complex urban environments. Understanding this value not only helps us understand the upper limit of the performance of traditional CFD methods but also provides a clear target for comparison with our proposed new method, underscoring the significance of our research.

In all experiments below, the output time step interval is 0.1 seconds, with the first output time step (t1) corresponding to the 181st step in the test data, and subsequent time steps added sequentially.

4.1 Comparison of Model Trained on Patches and Whole Urban Area

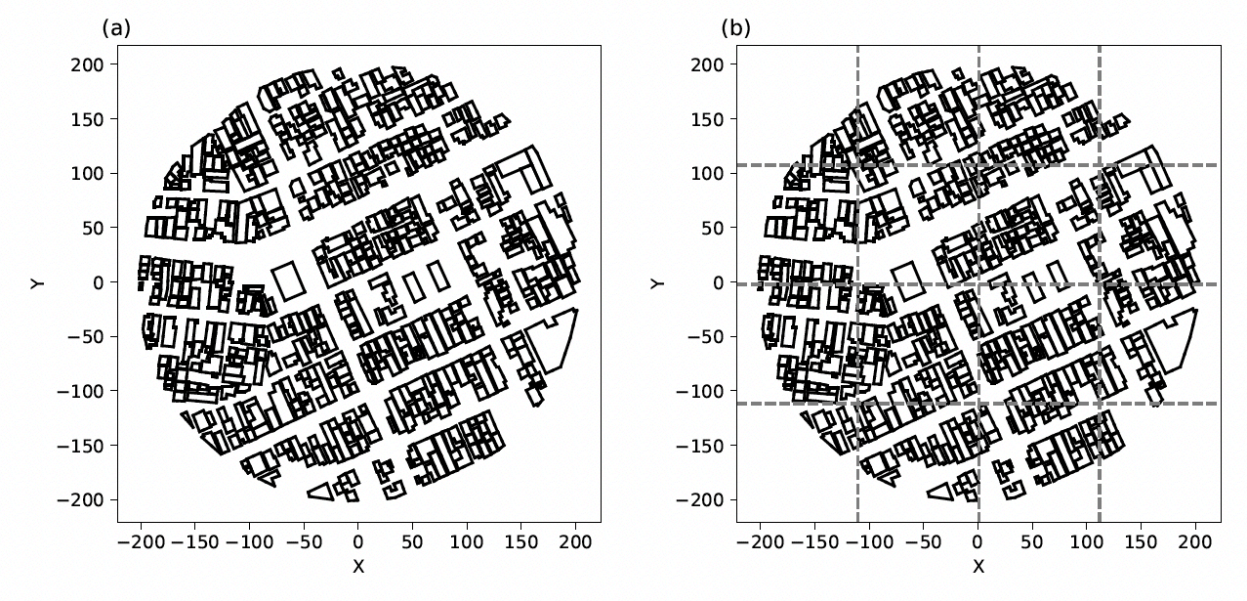

Figure 4.2 (a). The original whole block layout. (b). The layout segmented into smaller blocks for FNO-based generalization tasks.

The wind field data division pattern utilized in the two distinct training methodologies are illustrated in Figure 4.2(a) and Figure 4.2(b), respectively. Figure 4.2(a) depicts the comprehensive wind field training method, which employs the wind field data from the entire city. In this manner, the model is able to simultaneously observe the wind speed distribution across the entire city and the impact of buildings on the wind field. Figure 4.2(b) depicts the training method that utilizes smaller wind fields, wherein the city wind field is partitioned into numerous discrete local regions, each comprising local information about the wind field. The model is trained in these local areas to enhance its capacity to capture local details. This comparison elucidates the distinction between the two training methods: the complete wind field training method prioritizes global information, whereas the small patches training method emphasizes local precision.

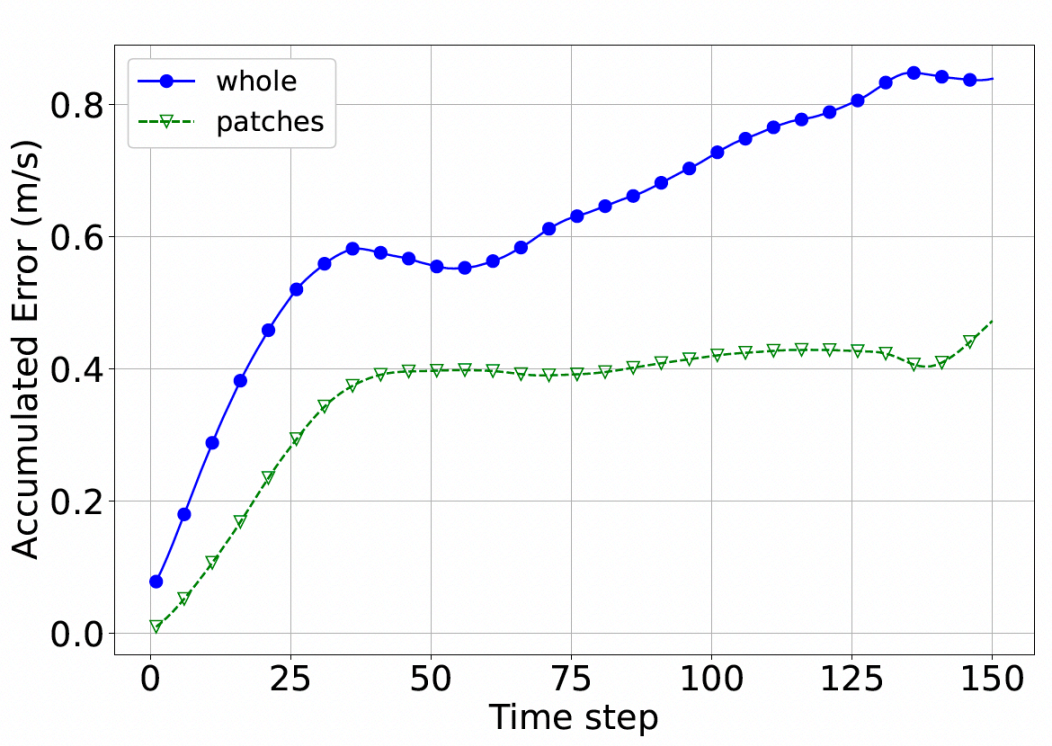

Figure 4.3 Comparison of accumulated average absolute error in Niigata over time between the models trained on the whole wind field and the one trained on 64 * 64 patches wind field.

Figure 4.3 illustrates the absolute prediction errors of the FNO model trained on patches wind field and the FNO model trained on the whole urban wind field in the T186-T255 Niigata north wind field. As illustrated in Figure 4.3, the prediction errors of the FNO model trained on patches wind field in the T186-T255 time period are less than those of the FNO model trained on the whole urban wind field, with the error being less than 0.5 m/s. The results demonstrate that the FNO model trained on patches wind field has reached the same level of prediction accuracy as that of mainstream CFD software like Fluent and OpenFoam, with an absolute prediction error of 0.5 m/s. In contrast, the FNO model trained on the whole urban wind field exhibits an error greater than 0.5 m/s after the 25th step, with a faster accumulation of errors than that observed in the FNO trained on patches wind field. Based on the result, it can be concluded that training FNO with patches wind fields is preferable to training FNO with whole wind fields.

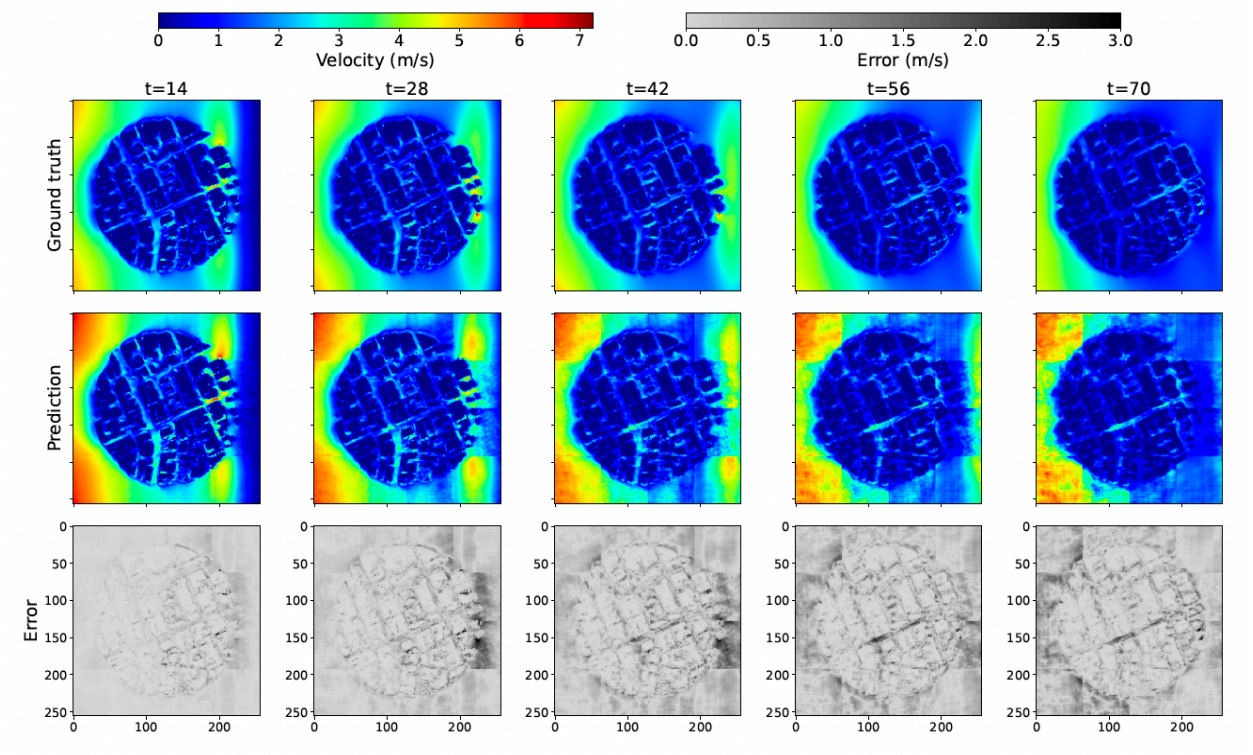

Figure 4.4 Prediction results of the Niigata wind field at five selected time steps (t = 14, 28, 42, 56, 70) using the FNO model trained on 64 * 64 urban patches wind field.

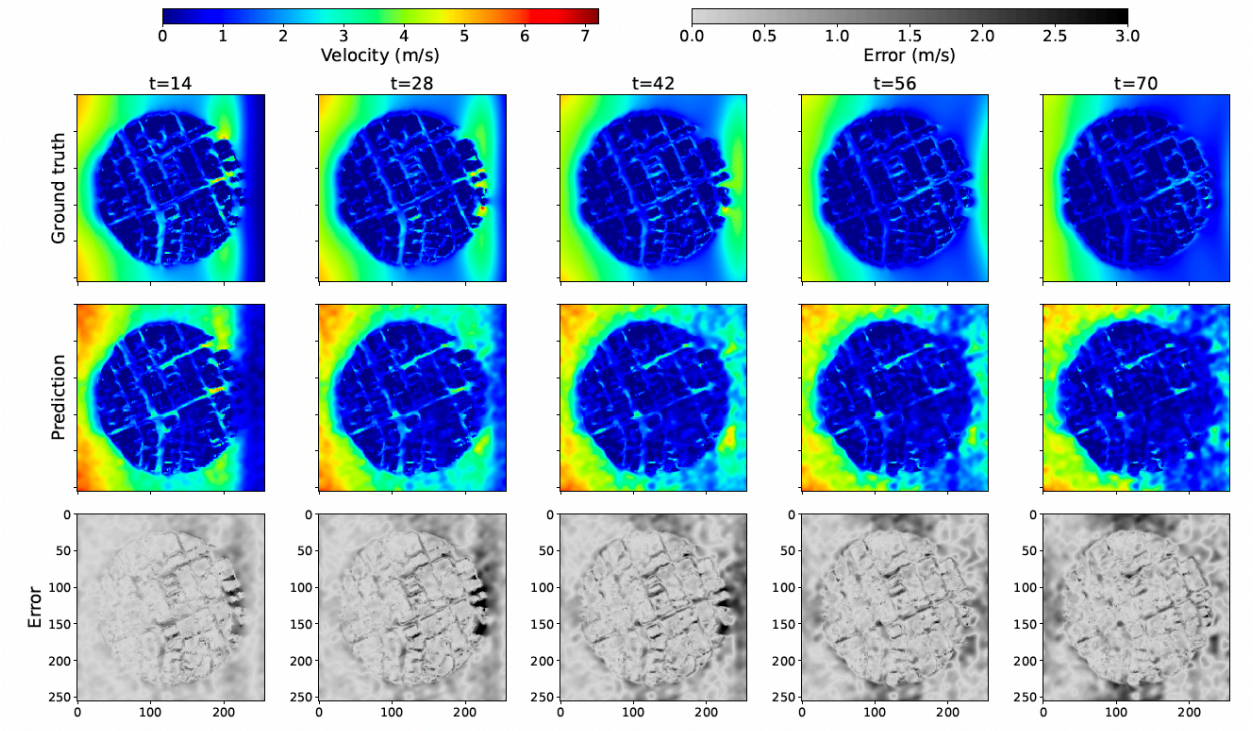

Figure 4.5 Prediction results of the Niigata wind field at five selected time steps (t = 14, 28, 42, 56, 70) using the FNO model trained on whole urban wind field.

Figure 4.4 and Figure 4.5 illustrate the performance of the FNO model trained on patches wind field and the FNO model trained on the whole wind field at 70 steps of prediction. Each figure depicts the actual wind speed field, the model-predicted wind speed field, and the error map between the two at distinct time points (t=14, t=28, t=42, t=56, t=70), arranged in columns. The initial row depicts the actual wind speed field. The flow of wind speed between buildings gives rise to a complex pattern, particularly at the periphery of the wind field and in areas characterized by a high density of buildings, where the speed gradient is pronounced. The second row depicts the prediction outcomes of the FNO model. At the initial time steps (t=14 and t=28), both the prediction results of the small-block training model and the whole-block training model exhibit a closer alignment with the actual values. However, as the time step increases, particularly at t=56 and t=70, the prediction results begin to diverge from the actual values, particularly at the edges of the wind field and in areas with complex geometries. The third row depicts the discrepancy between the predicted and actual wind speed fields. The discrepancy between the two figures is relatively minor at the outset but progressively accumulates with the passage of time. The model trained with small patches (Figure 4.2(a)) exhibited a lower error rate in long-term step prediction than the model trained with a whole patch (Figure 4.2(b)), particularly at time steps 56 and 70. The model trained with a whole patch exhibits a more pronounced error, particularly in regions with dense urbanization and at the periphery of the wind field. In conclusion, the model trained with small wind field segments demonstrates superior performance in capturing the local characteristics of the wind field and long-term step prediction. Conversely, the model trained with the whole wind field exhibits larger errors in long-term step prediction, particularly in complex geometric areas.

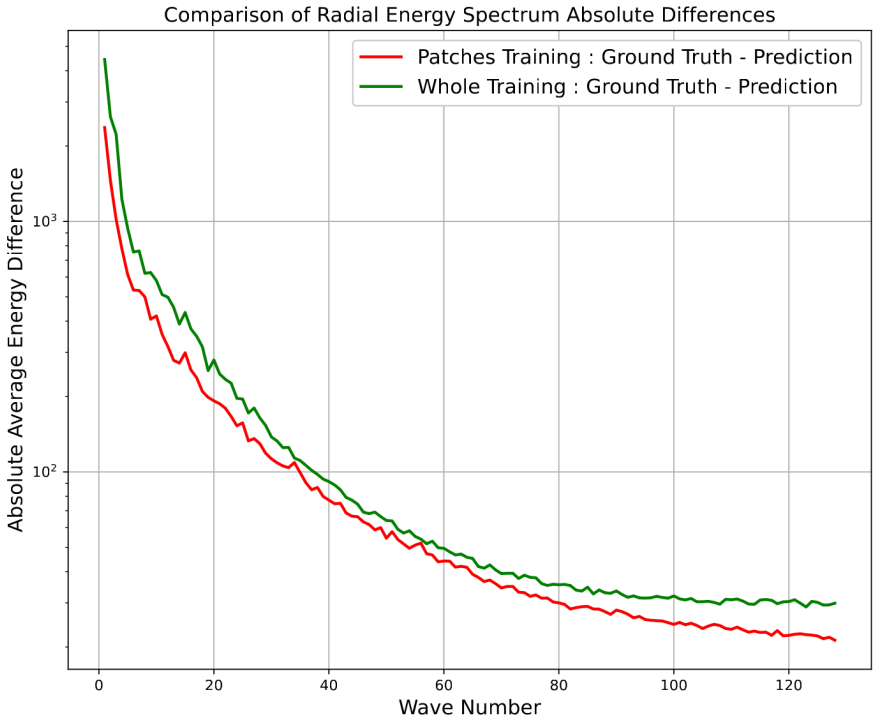

Figure 4.6 Comparison of the radial energy spectrum absolute differences between the Ground Truth and models trained on Patches (red) and Whole Training (green).

Figure 4.6 shows a comparison of the absolute mean energy difference. The red line represents Ground Truth vs Patches Training, and the green line represents Ground Truth vs Whole Training. As can be seen from the figure, in the region of small wave numbers, the difference between the two is relatively small, but as the wave number increases, the energy difference of the whole training increases significantly, while the small patch training maintains a relatively small energy difference. This further shows that small patch training can better approximate the true energy distribution at high wave numbers (small scales). In summary, the analysis in the figure shows that small patch training is superior to whole patch training in capturing the energy distribution of small-scale turbulence.

4.2 Influence of the SDF Data on Urban Wind Field Prediction

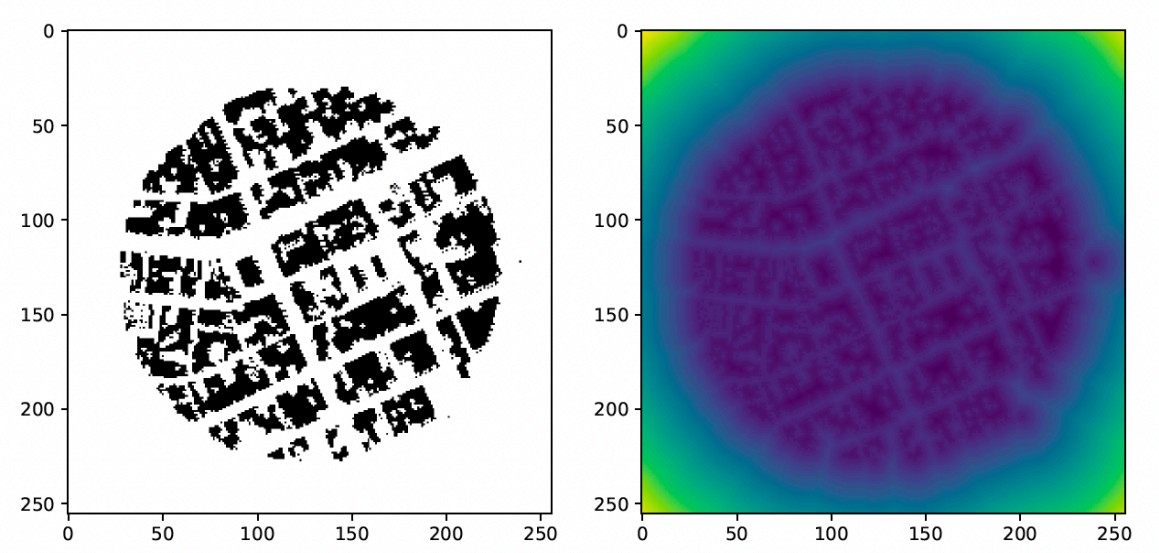

Figure 4.7 Visualization of the building layout(left) and corresponding Signed Distance Function (SDF)(right) for the Niigata urban environment.

In Figure 4.7, the map on the left depicts the distribution of buildings in an urban area. Black areas indicate the presence of buildings, while white areas represent open spaces or non-buildings. The map has been processed using a binary method, with the objective of clearly identifying the contours and distribution of buildings in the city. The resulting SDF (signed distance function) map, generated based on the aforementioned image, is presented on the right. The coloration gradually transitions from the center to the periphery, with purple indicating areas situated at a greater distance from the edifices and green and yellow indicating areas in closer proximity to the buildings. The SDF map reflects the distance of each pixel from the nearest

building boundary. The gradual change in color allows for the visualization of the relative distance distribution of different locations in space from the buildings.

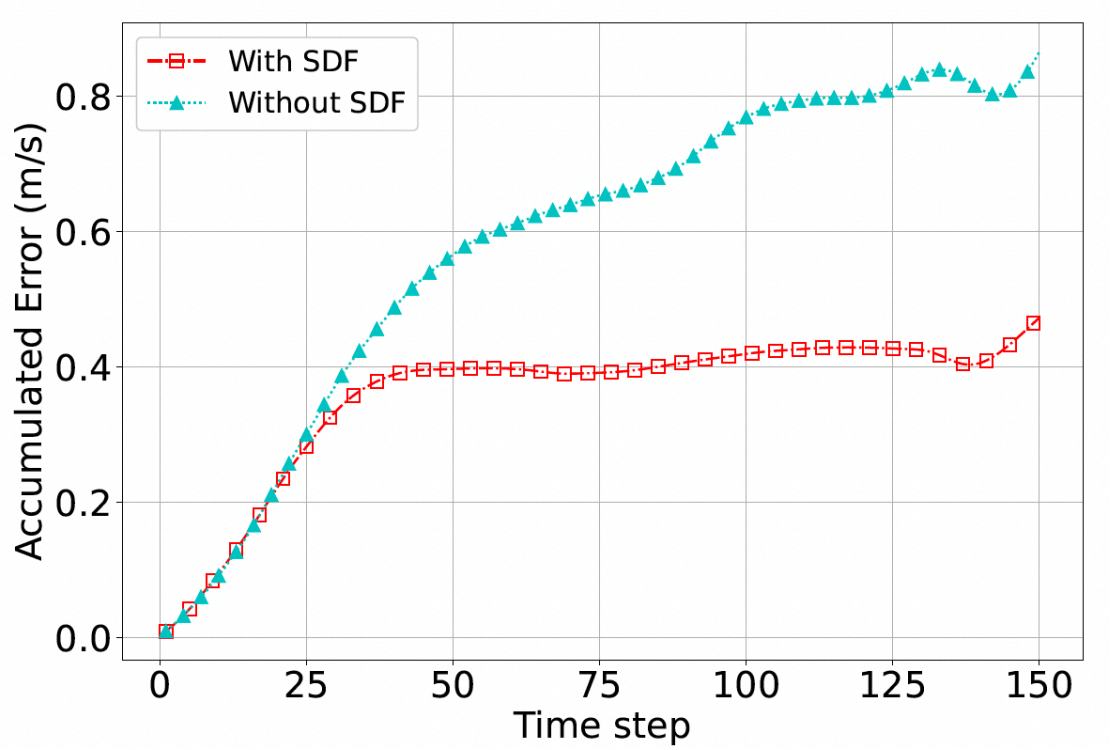

Figure 4.8 Comparison of accumulated error over time between the models trained With SDF (red line with squares) and Without SDF (cyan line with triangles).

Figure 4.8 illustrates the absolute prediction errors of the FNO model trained with SDF data and the FNO model trained without SDF data in the T1-T150 Niigata North wind field. The data utilized for training were derived exclusively from the Niigata West wind field. As illustrated in the figure, the prediction error of the FNO model trained with SDF data is less than 0.5 m/s during the T1-T150 time period. In contrast, the error of the FNO model trained without SDF data exceeds that of the FNO model trained with SDF data at approximately T30 and continues to increase. In light of these findings, it can be concluded that the FNO model trained with SDF data outperforms the FNO model without SDF data under identical experimental conditions. The incorporation of SDF data enhances the spatial correlation between data points, thereby enabling FNO to extract a greater number of physical characteristics and to achieve more stable prediction outcomes.

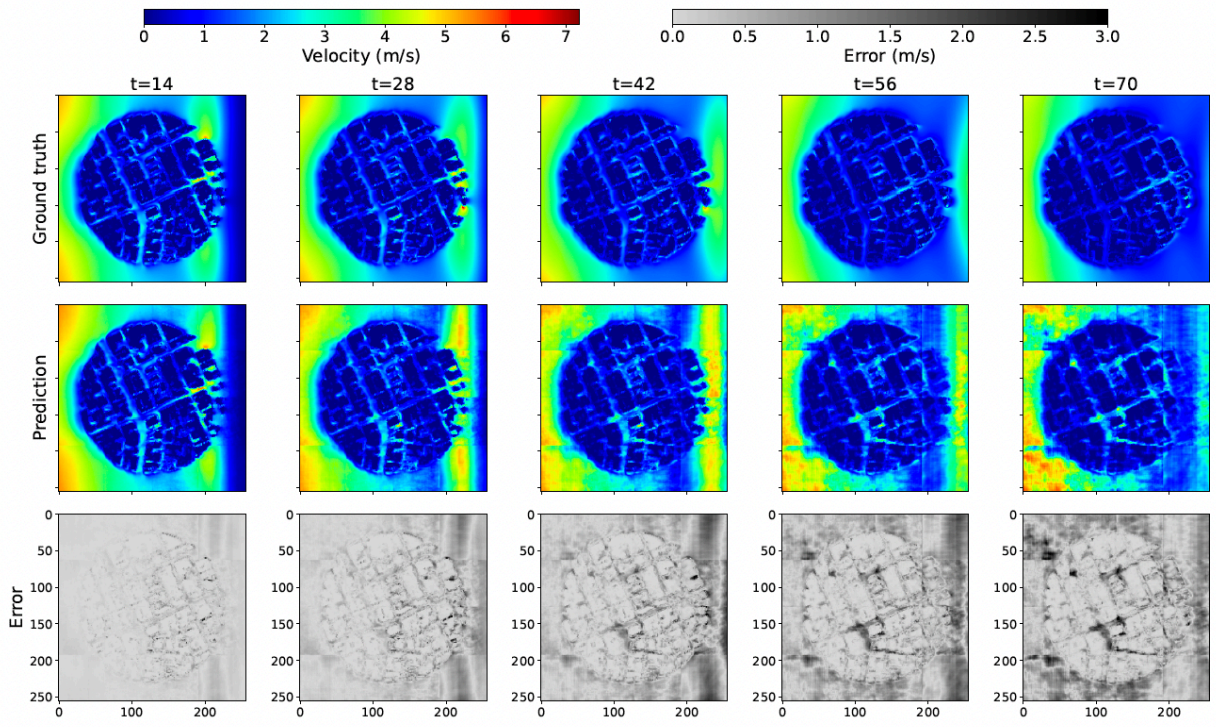

Figure 4.9 Prediction results without SDF at five selected time steps (t = 14, 28, 42, 56, 70).

Figure 4.9 illustrates the outcomes of a 70-step prediction utilizing a FNO model that was trained with a limited wind field on the North Wind test set, without the incorporation of SDF data. The figure illustrates the true value, the predicted value, and the discrepancy between the two at various time points (t=14, t=28, t=42, t=56, t=70) for each column 1. The true value (presented in the first row) illustrates the true wind speed distribution at varying time points. The regions exhibiting elevated wind speeds are concentrated in the flow channels between the edifices, particularly in the vicinity of the edges, which evinces intricate fluid dynamics in these areas. The predicted value is presented in the second row. The model's prediction is more closely aligned with the actual wind speed field at the outset (t=14, t=28). However, as the time step increases, the discrepancy between the predicted and actual values widens, particularly at t=56 and t=70, where the prediction deviation in the vicinity of the building edges is pronounced. The error field is presented in the third row. As time progresses, the model's errors accumulate gradually, particularly in the vicinity of complex geometric boundaries, where the error values increase significantly. This further indicates that the FNO model, which does not use SDF data, has certain limitations when dealing with complex geometric structures and capturing physical relationships, particularly in the prediction of longer time steps, where the error diffusion is more pronounced. In conclusion, the model that was trained using the small-scale wind field without SDF data demonstrated a slight decline in performance when evaluated on the North Wind test set. While the initial predictions were relatively accurate, the error increased significantly with the passage of time. The absence of SDF data impeded the model's capacity to accurately delineate the spatial relationships and geometric features intrinsic to the wind field, thereby compromising the overall accuracy of the prediction.

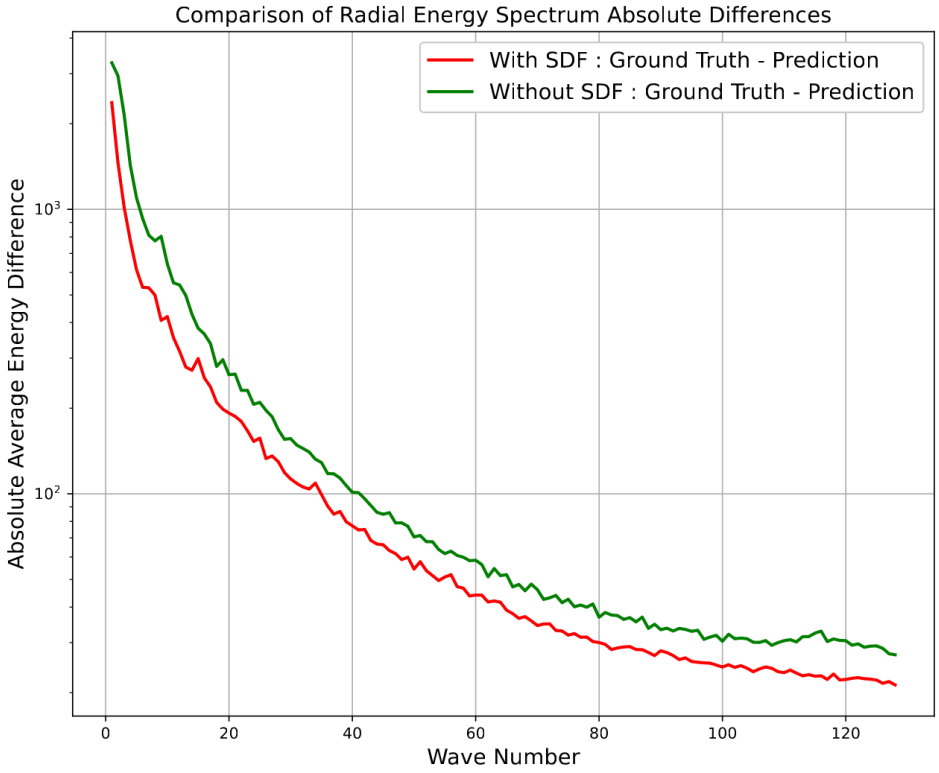

Figure 4.10 Comparison of radial energy spectrum absolute differences between Ground Truth vs With SDF (red) and Ground Truth vs Without SDF (green).

Figure 4.10 shows a comparison of the absolute difference in radial energy spectra trained with and without SDF. The vertical axis of the figure represents the absolute mean energy difference, and the horizontal axis is the wave number, which represents the change from large scale to small scale. The red line represents Ground Truth vs With SDF, i.e. the absolute energy difference between the model and the real data when SDF data is involved in training. The green line represents Ground Truth vs Without SDF, i.e. the absolute energy difference between the model and the real data when SDF data is not involved in training.

As can be seen from the figure, the energy difference between training with SDF (red line) and without SDF (green line) is generally lower in all wave number ranges, especially in the low wave number (indicating large-scale structure) region. As the wave number increases (entering the small-scale region), the difference between the two gradually decreases, but overall the model with SDF still performs better. This shows that the introduction of SDF data significantly improves the model's accuracy in predicting the energy distribution of the wind field, especially when capturing large-scale turbulent structures. SDF provides more physical information, bringing the model closer to the real energy distribution.

4.3 Hyperparameters Researchs

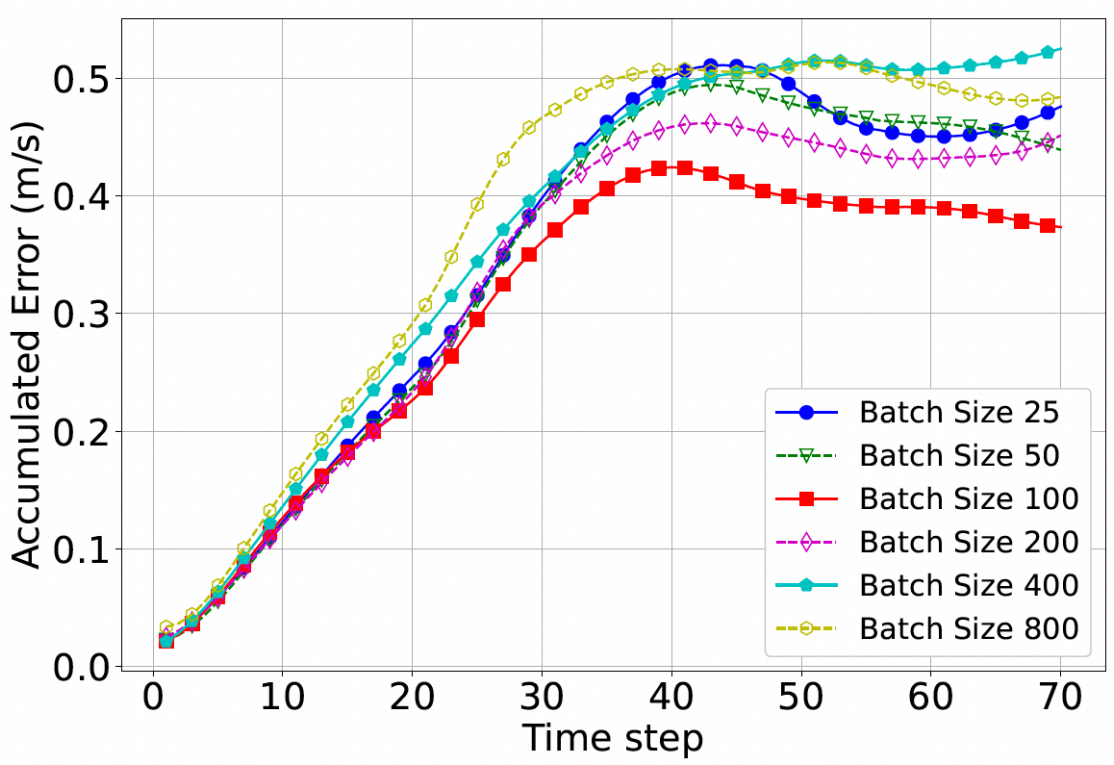

Figure 4.11 Accumulated average absolute error in Niigata over time for wind field predictions with different batch sizes.

Figure 4.11 illustrates the experimental results of wind field prediction at a specified time point for 70 consecutive time steps, and evaluates the impact of different batch sizes on the cumulative prediction error. Each curve represents the trend of the prediction error with respect to the time step for different batch sizes under the same model settings. As illustrated in Figure 17, the model with a batch size of 100 demonstrates the most optimal performance throughout the time period. The cumulative error remains relatively stable during the 70 steps of prediction, and the error maintains the lowest level compared with other batch sizes' result. This demonstrates that a batch size of 100 represents an optimal balance between the frequency of gradient updates and the model's predictive performance. The use of smaller batch sizes (e.g., 25 and 50, indicated by the blue and green lines, respectively) results in lower initial prediction errors. However, the cumulative errors increase rapidly with the number of time steps and exhibit greater fluctuations in long-term predictions than batch size 100. This phenomenon may be attributed to the fact that smaller batch sizes prompt the model to update the gradient during prediction, which results in a gradual accumulation of errors in long-term predictions. Conversely, while the batch sizes of 400 and 800 result in lower errors in the initial predictions, the cumulative errors increase larger than batch size 100 with the increase of the time step. Notably, the batch size of 800 exhibits a pronounced surge in errors pertaining to long-term

predictions. A larger batch size may result in a lower update frequency during model training, thereby making it challenging to adjust the prediction error in a timely manner, particularly in the later stages of the prediction, where the impact is pronounced.

In contrast, although the batch size of 200 shows superior performance compared to 400 and 800, its accuracy is not significantly improved compared to the batch size of 100, and it requires more GPU resources. In conclusion, the batch size of 100 can effectively balance the gradient update of the model and the prediction accuracy, resulting in enhanced generalization ability and a reduced cumulative error in long-term wind field prediction.

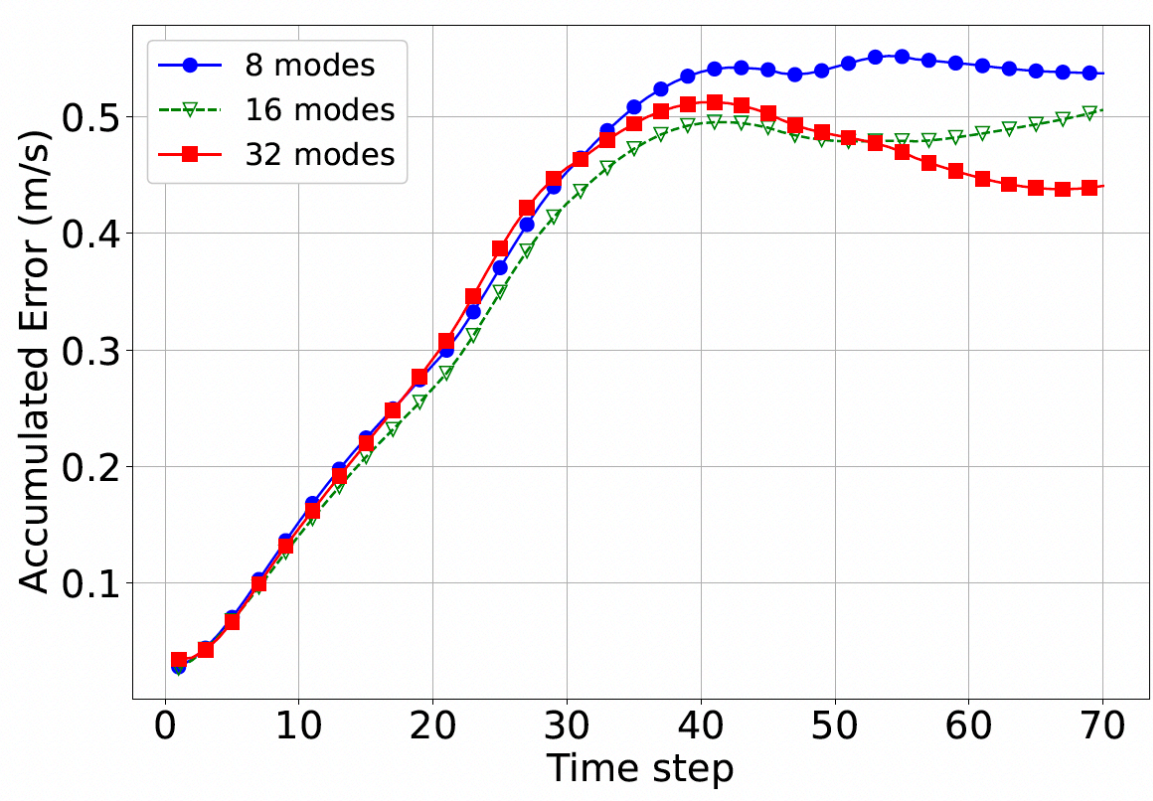

Figure 4.12 Accumulated average absolute error in Niigata over time for wind field predictions with different modes.

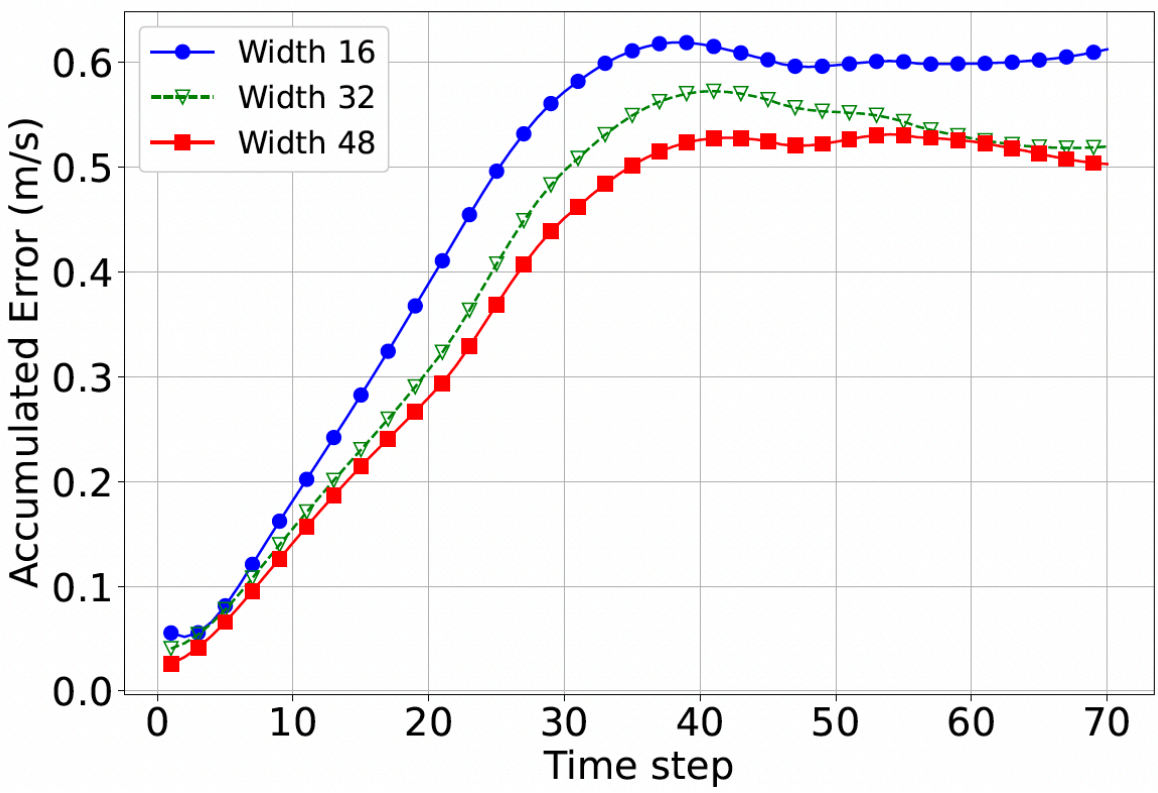

Figure 4.13 Accumulated average absolute error over time for wind field predictions with different width (convolutional kernel size).

The objective of this study was to evaluate the impact of varying settings for the modes and width parameters of the FNO model. The modes parameter denotes the number of modes extracted by the FNO model following a Fourier transform of the input data. In contrast, the width parameter represents the size of the convolution kernel employed by the FNO model during internal linear transformation. The results demonstrate that augmenting these two hyperparameters has a considerable, beneficial impact on the model's performance.

In Figure 4.12, the influence of varying the number of modes was investigated, with values ranging from 8 to 32. Given that the dimensions of the input patch are 64 * 64, the maximum number of modes that can be obtained is 32. The results demonstrate that as the number of modes increases, the model's predictive accuracy improves gradually. This is presumably due to the fact that a greater number of modes allows for a more comprehensive capture of the intricate patterns and characteristics inherent in the data, thereby enhancing the model's expressiveness. However, an excessive number of modes can result in a considerable increase in computational cost. Therefore, it is essential to achieve a balance between performance and computational resources. The utilization of 128 modes on a complete block of 256 * 256 is not currently supported by our GPU; And the largest mode we can choose for 64x64 patches is 32. Thus, 32 modes were employed instead. Figure 4.13 illustrates that an expanded width enables the FNO model's convolutional kernel to discern intricate data structures with greater precision, thereby

enhancing prediction accuracy. However, an increase in width is also accompanied by an increase in training time and memory consumption. Therefore, although a larger width performs better in terms of accuracy, it is necessary to consider the consumption of computing resources in practical applications. We selected 48 as the maximum width that our GPU resources can accommodate within a tolerable usage time.

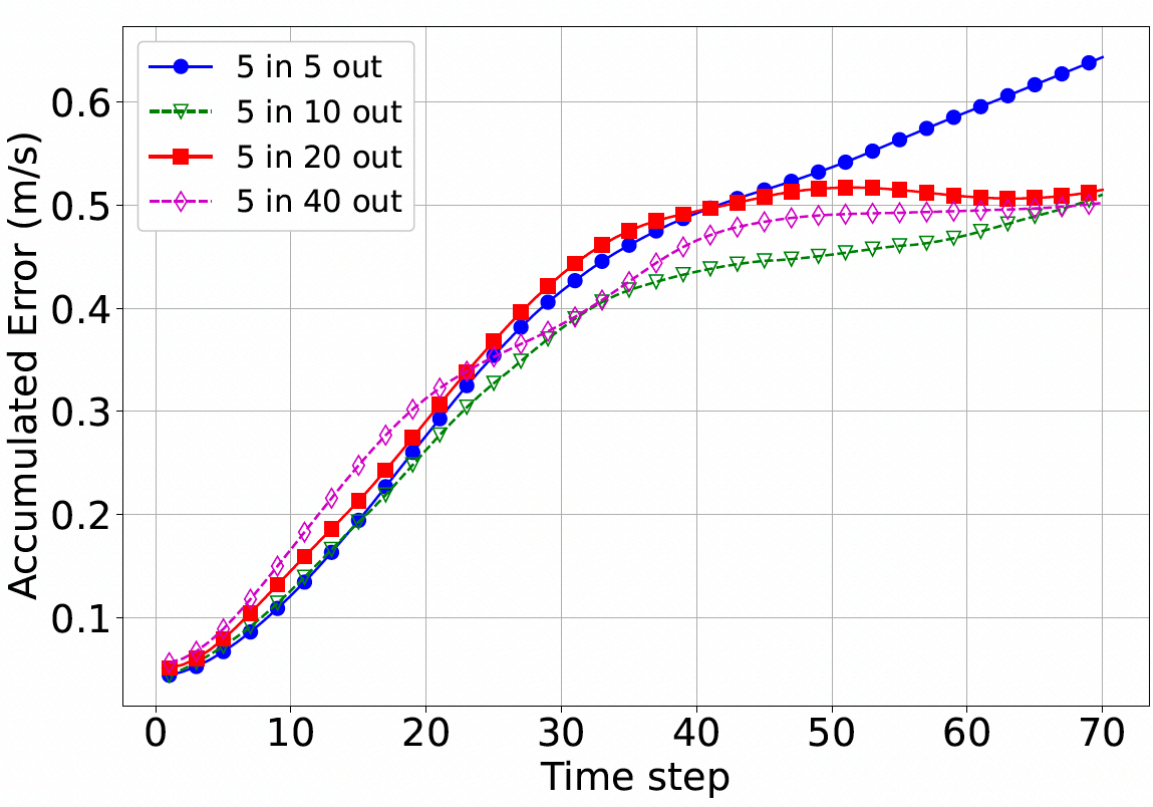

Figure 4.14 Accumulated average absolute error in Niigata over time for wind field predictions with different output in one iteration.

The FNO model is a single input-output model. Accordingly, in order to enhance the efficacy of the FNO model in forecasting wind patterns, it is essential to ascertain the optimal number of input-output time steps within a single iteration. An experiment was designed to test the input-output strategy employed in a single iteration. The impact of varying output step lengths (1, 10, 20, 40) was evaluated under the same input length (5 time steps). The results demonstrate that the 10-step output setting exhibits the optimal performance in terms of model efficacy.

As illustrated in Figure 4.14, when the output step is 1, although the short-term prediction error is minimal, the overall model exhibits reduced efficiency and suboptimal long-term prediction. This is due to the fact that excessive iterative updates result in a cumulative error that is repeated multiple times. Furthermore, the implicit equation associated with a multiple-step input and a single-step output is inherently more complex, ultimately leading to a reduction in

prediction accuracy within a single iteration. When the output step is 10, the model is able to significantly reduce the number of iterations while still outputting the same 70-step wind field. Furthermore, it has achieved a good balance between efficiency and prediction performance, particularly in medium- and long-term predictions (more than 10 steps). The existing data and FNO model parameter settings have been found to result in a significant increase in prediction error when the output step lengths are set to 20 and 40, particularly larger than outputting 10 steps in one iteration, despite a reduction in the number of iterations. An excessively long output step length in a single iteration impairs the FNO model's ability to effectively capture finer-grained wind field changes, resulting in a decline in prediction accuracy.

By comparing the same input (5 steps) with different output step lengths, we found that an output step length of 10 can achieve the optimal balance between model efficiency and prediction accuracy. This provides a valuable reference for selecting an appropriate iterative strategy for future practical applications.

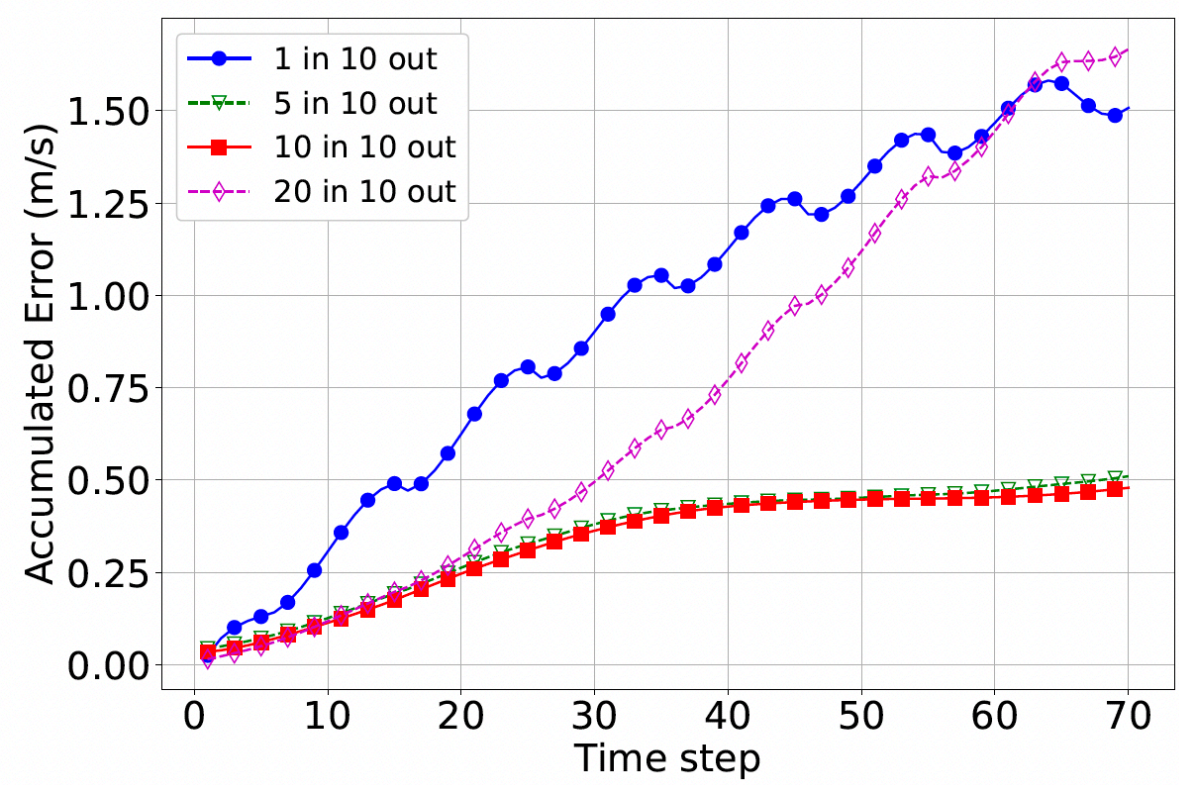

Figure 4.15 Accumulated average absolute error in Niigata over time for wind field predictions with different input in one iteration.

After determining that the 10-step output is the optimal setting, further tests were conducted to assess the impact of different input step sizes (1, 5, 10, 20) on model performance while maintaining the output step size at 10. As shown in Figure 4.15, the configuration with 5 input steps and 10 output steps in a single iteration exhibited the highest prediction accuracy. As illustrated in the figure, the prediction results for the 1-step input were significantly worse compared to the other three input configurations. This is because the 1-step input provides

insufficient feature information, preventing the FNO model from fully capturing the time series characteristics necessary for wind field prediction. In contrast, the 5-step input achieved optimal prediction results with minimal memory usage, particularly in capturing the complex temporal variations of the wind field. Although the prediction accuracy of the 10-step input is comparable to that of the 5-step input, the 10-step input requires more memory and computing resources. Further increasing the input step size did not significantly improve model performance, indicating that, in this case, additional input steps introduced a degree of information redundancy, with limited improvement in the FNO model's wind field prediction accuracy. Additionally, the experiments found that under the 20-step input configuration, prediction accuracy declined after the 20th output step compared to the 5-step input configuration. This could be attributed to the additional input steps increasing the complexity of the data relationships the model must learn, thereby reducing the overall learning efficiency of the FNO model. In summary, the optimal configuration for the FNO model in a single iteration is 5 input steps and 10 output steps.

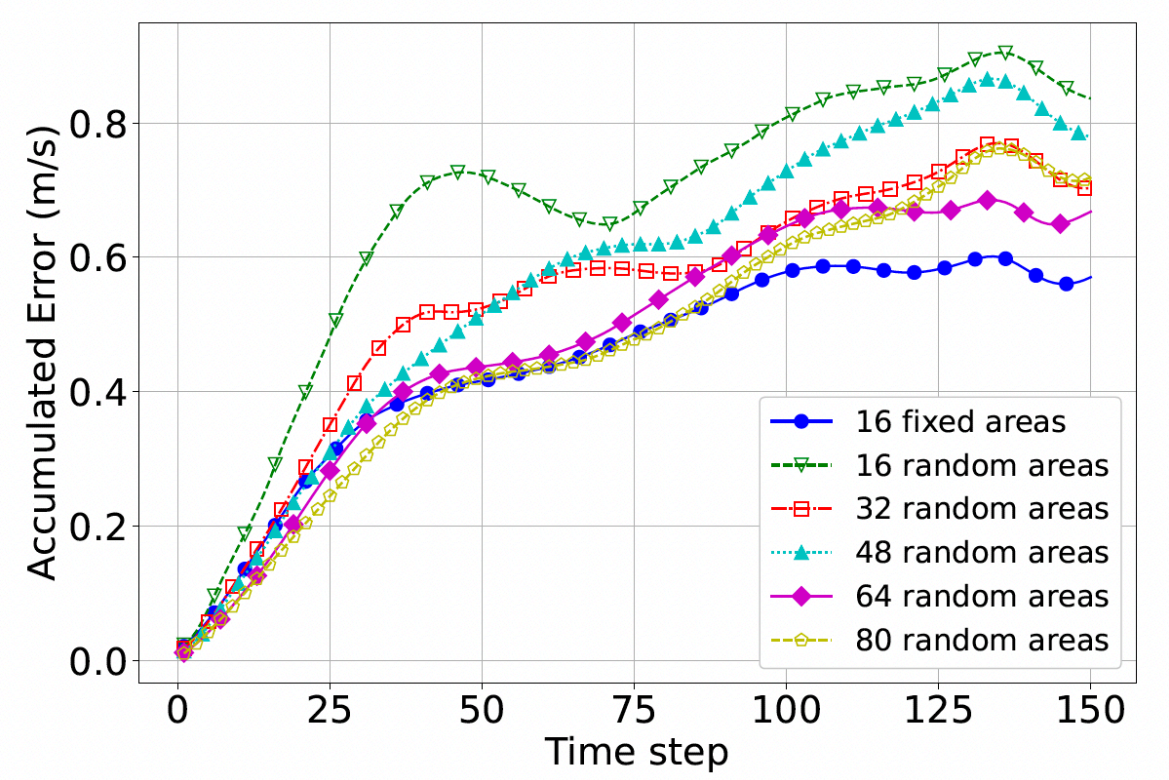

Figure 4.16 Accumulated average absolute error in Niigata over time for wind field predictions comparing 16 fixed areas (blue line) and different configurations of random areas (dashed lines).

To further substantiate the efficacy of our segmentation method (dividing the whole urban area into 16 patches which cover the whole urban area completely), a comparative analysis was conducted with the random segmentation method displayed by Figure 4.16. In the random

segmentation method, 16, 32, 48, 64, or 80, 64 * 64 patches were randomly selected. However, the experimental results demonstrated that none of these randomly selected schemes exhibited the same efficacy as our segmentation method. Our segmentation method ensures comprehensive coverage of the entire 256 * 256 wind, and the overlapping regions of adjacent small blocks permit the model to capture continuous local features and boundary information. The comprehensive and consistent coverage of the data ensures the model's capacity for generalization across the entire dataset, thereby enhancing the accuracy of the predictions. The random tessellation method is subject to the following limitations: In the random tessellation method, although different numbers of small blocks are randomly selected (16, 32, 48, 64, or 80), the resulting coverage of each tessellation is not guaranteed due to the inherent randomness and uncertainty of the process. It is possible that some crucial regions within the wind field may not receive sufficient training, while other areas may be sampled excessively. This could impede the model's ability to learn the uniform characteristics of the entire wind field. The randomness inherent to this method results in an uneven distribution of data, which in turn affects the accuracy of the model in certain areas and diminishes the overall efficacy of the approach, particularly in comparison to our solution. The presence of data redundancy and asymmetric information: Despite the random selection of larger numbers of smaller patches (such as 48, 64, and 80), this method does not yield sufficient information improvement. In comparison to the systematic coverage of our segmentation method, randomly selected small patches may result in data redundancy in certain areas, leading to the repetition of unnecessary information. Conversely, other areas may lack sufficient training data, which ultimately affects the overall performance of the model.

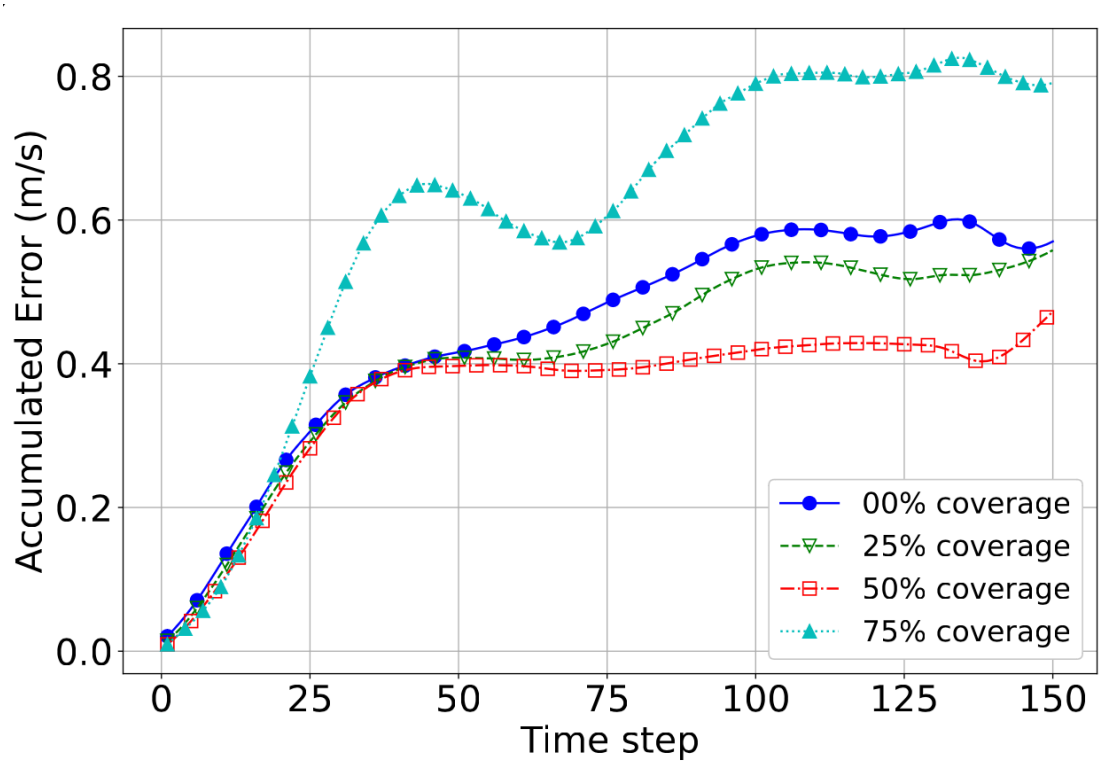

Figure 4.17 Accumulated average absolute error in Niigata over time for wind field predictions with different coverage levels: 00 coverage, 25 coverage, 50 coverage, and 75 coverage.

In the context of training the FNO in patches wind fields, it is essential to develop an effective and optimal small partition scheme. The 256 * 256 wind field data was divided into 64 * 64 small patches, with the division occurring from left to right and top to bottom. Four different step length schemes were then tested, with the step lengths varying from 64 to 16. The step lengths thus determine the degree of overlap between adjacent small patches, which in turn affects the data coverage and data volume. In the 64-step scheme, the split blocks are completely disjoint, resulting in a total of 16 64 * 64 patches. Although this method requires less training data and allows for more rapid model training, the lack of overlap between the small blocks makes it challenging for the model to capture the continuous information between boundaries, resulting in lower prediction accuracy. In comparison to the 64-step solution in Figure 4.17, the 48-step solution results in an increased overlap between the small patches, thereby increasing the amount of training data and enhancing the model's capacity to capture boundary information. However, the larger step size results in insufficient data coverage and no significant improvement in model performance. The results of our experiments indicate that the 32-step method produces the most optimal outcomes. At this juncture, the degree of overlap between each small block is considerable, the quantity of data following segmentation is augmented, and the model is able to discern the local characteristics and boundary information of the wind field from a more substantial corpus of training data, thereby exhibiting enhanced accuracy in prediction. This demonstrates that a moderate step size can achieve an optimal equilibrium between data coverage and model performance. Although the step size of 16 provides the greatest quantity of data, the overlap between each small patch is excessive, resulting in high data redundancy and, consequently, an adverse effect on the training efficiency of the model. The excessive overlap does not result in a notable enhancement in model performance and instead leads to an increase in computational overhead. The scheme with a step size of 32 was found to provide sufficient data coverage without excessively increasing data redundancy, thus achieving the optimal balance between performance and efficiency. This result provides a basis for optimizing data processing schemes for future urban wind field prediction models, especially when processing large-scale urban wind field data. A reasonable chunking step size can effectively improve model performance. This description clearly conveys the impact of the step size on model performance, explains why step size 32 is optimal, and provides a reasonable conclusion based on experimental results.

4.4 Performance of Model on Different Wind Directions

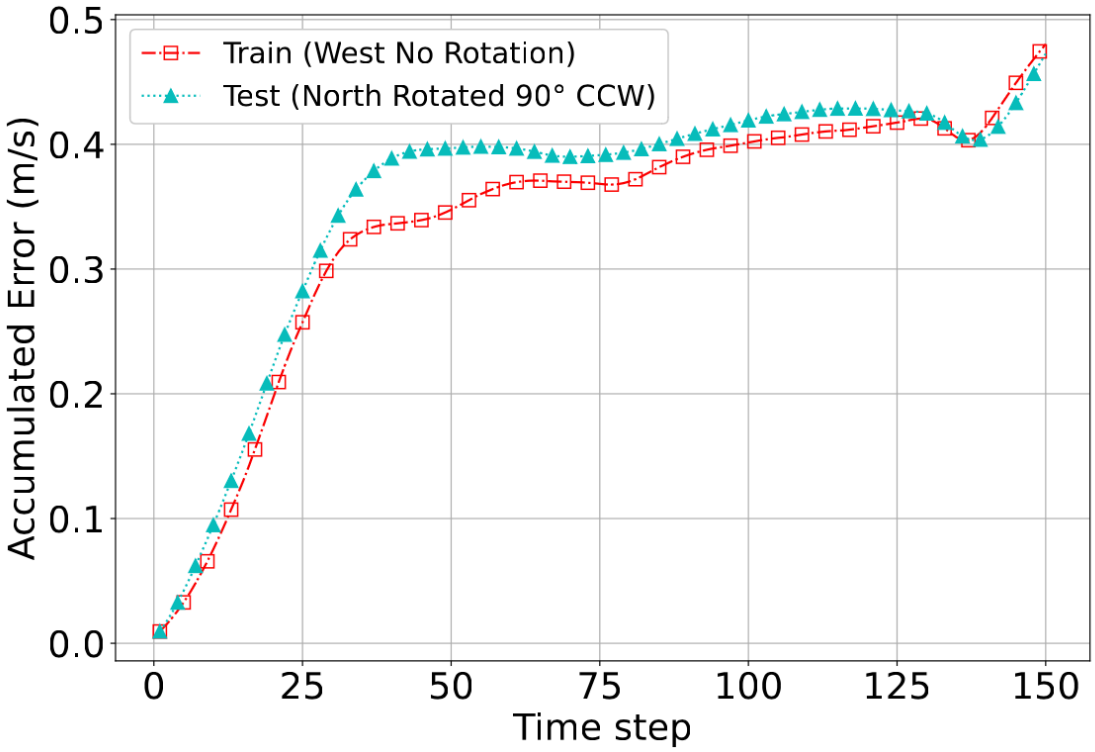

Figure 4.18 Accumulated average absolute error in Niigata over time comparing Train (West Original) and Test (North Rotated 90 degrees CCW, counter-clock wise).

Figure 4.18 above depicts the cumulative error of the FNO model (AI method) trained on the west direction Niigata urban wind field and subsequently generalized to the rotated Niigata north urban wind field. In the experiment, with the exception of a change in wind direction, the wind speed and urban geometry remained constant. The red line represents the performance of the FNO model on the West Wind training set, and the blue triangle line illustrates the error performance on the rotated North Wind test set. As illustrated in the figure, the FNO model demonstrates robust generalization capabilities with respect to the rotated North Wind test set. The cumulative error between the training and test sets is comparable between time steps T1 and T30, indicating that the model can maintain consistent prediction accuracy under varying wind conditions as it did under training conditions. Following T30, the discrepancy between the test and training sets is marginal, with the former exhibiting a slight elevation in error relative to the latter. However, this prediction absolute mean error remains within the 0.45 m/s range, which is in close alignment with the established benchmark for error in mainstream CFD software, such as Fluent (0.5 m/s). It is noteworthy that despite an increase in error when dealing with wind direction changes, the FNO model demonstrates comparable overall prediction accuracy to CFD methods within a limited time frame, while exhibiting a substantial advantage in computational

speed. In comparison to traditional CFD methods, the FNO model markedly reduces the computational time through the utilization of deep learning technology based on pure data-driven methods, thereby facilitating the generation of high-precision wind field prediction results within a specified output time frame. The experimental results demonstrate that the FNO model is not only capable of generating prediction results in a relatively short time frame, but also exhibits robust generalization capabilities. In particular, when the wind speed and urban geometric structure remain constant, the impact of wind direction changes on the model is minimal. In summary, FNO achieves a level of accuracy comparable to CFD in a shorter calculation time, thereby providing an efficient and accurate solution for complex wind field prediction. This provides further evidence of the potential applications of FNO in urban wind field simulation.

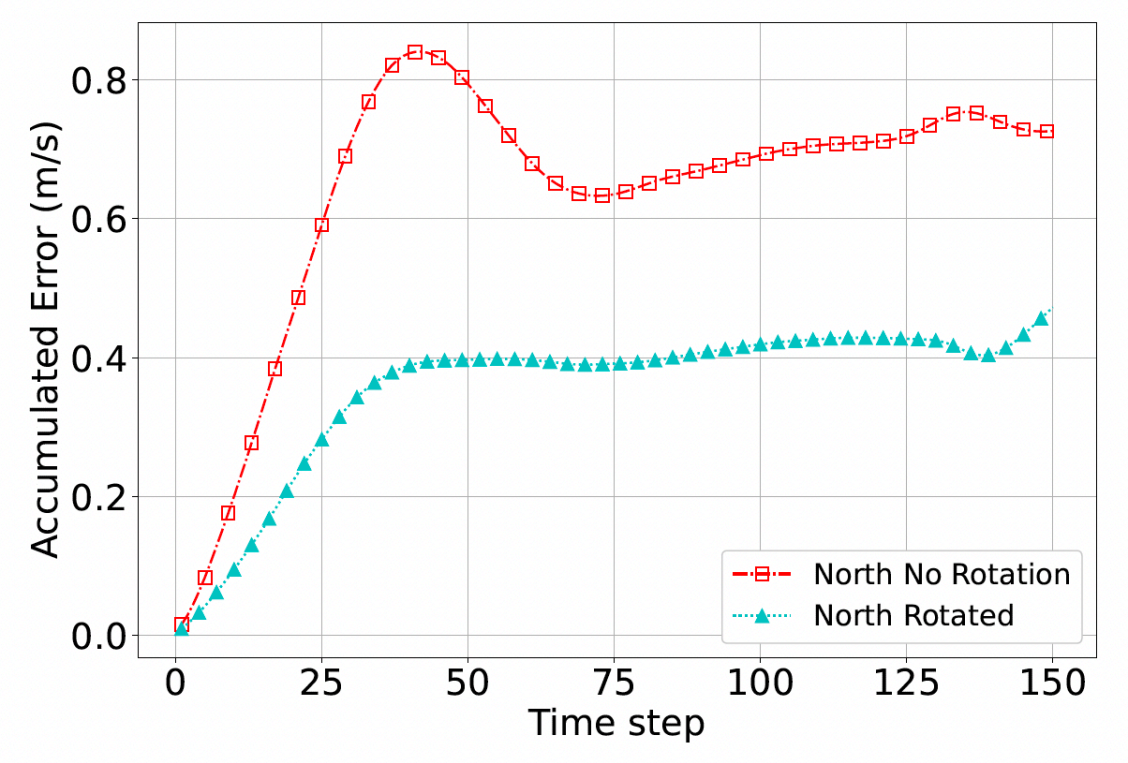

Figure 4.19 Accumulated average absolute error in Niigata comparison between predictions on North No Rotation Niigata data and the North Rotated Niigata data over 150 time steps.

Figure 4.19 illustrates the predictive outcomes of the FNO model in Niigata north wind urban field. A comparison of the cumulative error performance of the unrotated north wind (red line) with that of north wind rotated 90 degrees counterclockwise (blue triangle) is presented. The experimental results demonstrate that the prediction accuracy of the rotated northerly wind field is markedly superior to that of the unrotated wind field. The cumulative error of the rotated northerly wind remains below 0.4 m/s throughout the prediction process, whereas the error of the unrotated northerly wind rises rapidly after T40, reaching a peak value of over 0.8 m/s. The objective of this experiment is to rotate the northerly wind 90 degrees counterclockwise to create

a visual similarity to the westerly wind, thereby aligning it more closely with the training data of the model, which is based on the westerly wind. Despite the maintenance of constant wind speed and urban geometry, the unrotated north wind test set exhibited notable divergence from the training set in the spatial domain, resulting from the alteration in wind direction. This discrepancy impeded the model's capacity to generalize, leading to a pronounced surge in cumulative error. Following the rotation, the test set and training set exhibited a congruent wind direction, thereby enabling the FNO model to more effectively apply the features acquired during training to the test set, thereby enhancing prediction accuracy. The results of this experiment have significant implications for future research on the generalization of deep learning models. The results demonstrate that ensuring consistency between the visual wind directions of the test set and the training set is a crucial step in enhancing the generalization ability of the model when dealing with wind field prediction tasks. By modifying the wind direction, the test set can be aligned with the training set, thereby enhancing the accuracy of the predictions. This offers a novel perspective for future deep learning research, particularly in the context of addressing directional discrepancies between data sets. This strategy can be directly applied to other fluid dynamics or wind field prediction tasks, thereby enhancing the generalization performance of the model.

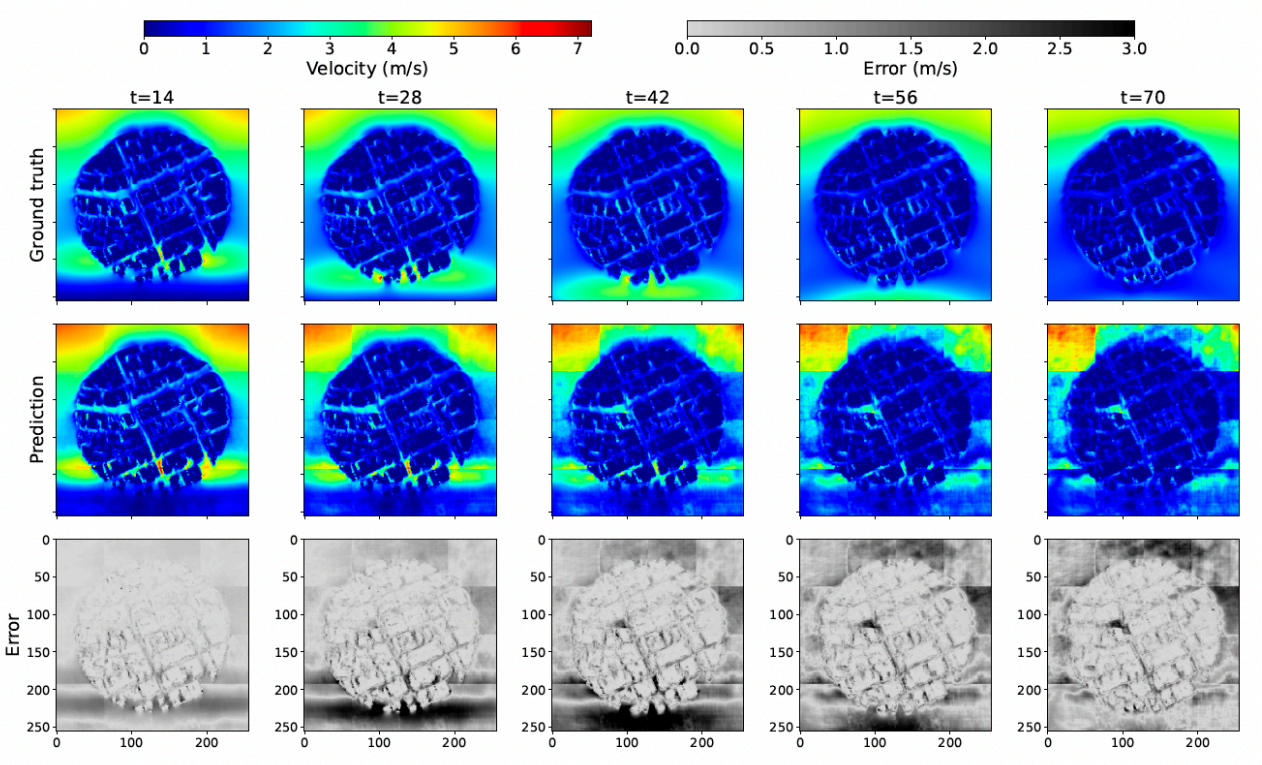

Figure 4.20 Comparison of ground truth, predicted results, and the corresponding error maps for the North wind simulation without rotation in Niigata.

Figure 4.20 illustrates the outcomes of a forecast conducted using a FNO model that was trained with a modest wind field situated in a northerly wind field. The experiment did not entail a rotation of the northerly wind field. The figure depicts the true value, the predicted value, and the discrepancy between the two at various time points (t=14, t=28, t=42, t=56, t=70) for each column.

The true value (presented in the first row) depicts the authentic wind speed distribution at varying time points. The velocity of the fluid forms a more complex distribution in the channel between the buildings, particularly in the marginal area of the wind field, which exhibits a pronounced velocity gradient.

The second row displays the predicted values. At the initial time steps (t=14, t=28), the model's prediction in close agreement with the true values. Nevertheless, as the time step increases, the model's prediction error gradually increases in complex areas of the wind field (such as the edges of buildings and low wind speed areas), particularly at t=56 and t=70, exhibiting significant deviations. This suggests that the FNO model is challenging to maintain stability in long-term prediction without rotation. The error field (third row) illustrates the distribution of discrepancies between the predicted and actual values. Although the initial error is minimal, the error in the vicinity of the building increases markedly with the prolongation of the time step, particularly at t=56 and t=70.

Furthermore, the accumulation of error is more pronounced, particularly in the peripheral regions of the wind field. This suggests that, in the absence of rotation, the model is unable to effectively capture the global characteristics of the northerly wind field. In conclusion, the unrotated northerly wind field prediction demonstrates that the model exhibits suboptimal performance in complex geometric regions and over extended time steps, with errors accumulating over time. These findings further suggest that appropriate geometric transformations (such as rotation) may play a pivotal role in enhancing the model's generalization capacity in diverse wind directions.

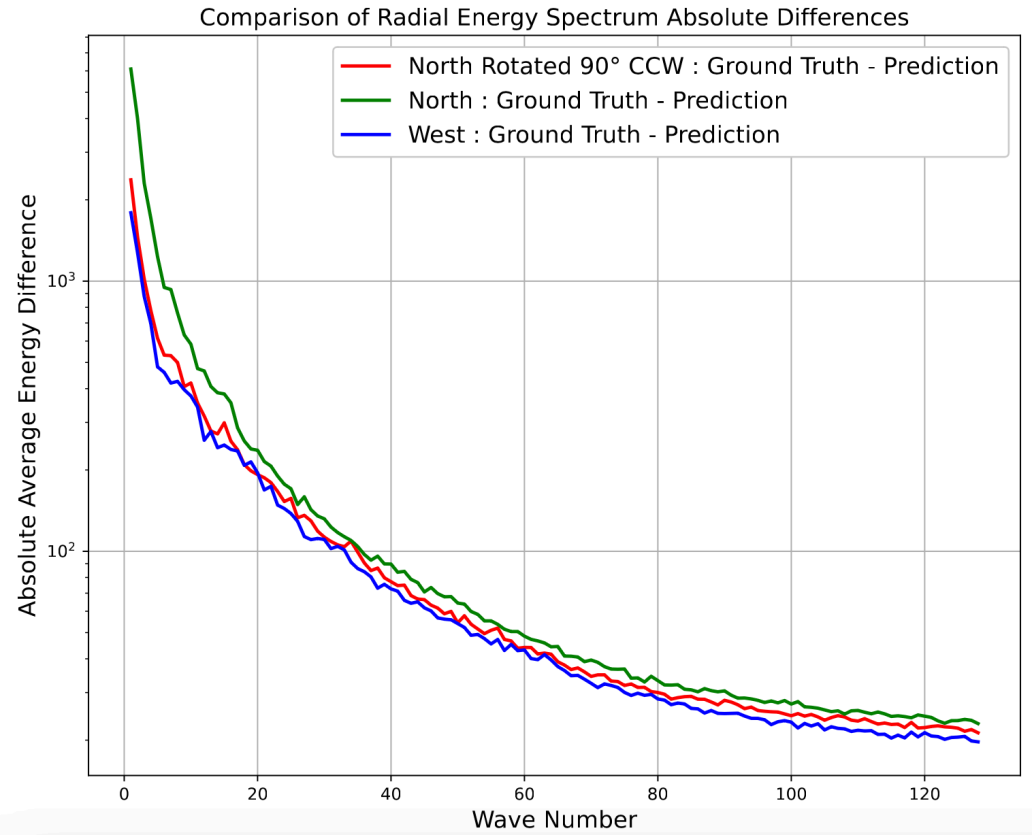

Figure 4.21 Comparison of radial energy spectrum absolute differences in different wind directions

Figure 4.21 shows a comparison of the absolute differences between the radial energy spectra of the wind field prediction and the ground truth for different wind directions. The vertical axis is the absolute mean energy difference, and the horizontal axis is the wave number, which indicates the structure from large to small scales.

The red line represents North Rotate Ground Truth vs North Rotate Prediction, i.e. the difference between the ground truth and the prediction after the north wind has been rotated. The green line represents North No Rotate Ground Truth vs North No Rotate Prediction, i.e. the difference between the ground truth and the prediction after the north wind has not been rotated. The blue line representsWest No Rotate Ground Truth vsWest No Rotate Prediction, i.e. the difference between the ground truth and the prediction after the west wind has not been rotated.

As can be seen from the Figure 4.21, in the low-wave number region (large-scale structure), the energy difference of the green line is significantly higher than that of the red and blue lines, indicating that the unrotated northerly wind model has a larger error in large-scale prediction. However, the difference between the rotated northerly wind model (red line) and the westerly wind model (blue line) is relatively small, indicating that the rotated northerly wind prediction results have a smaller error in the large-scale structure and are closer to the westerly wind prediction results. As the wave number increases (small-scale structure), the difference in

energy between the three lines gradually decreases and basically converges in the high-wave number region, indicating that the effect of rotation processing on the prediction error is relatively small at small scales. There is not much difference in prediction error between the rotated northerly wind model and the unrotated northerly wind model, as well as the westerly wind model.

Overall, the prediction of the rotated northerly wind model is closer to the true value in terms of large-scale structure, while the unrotated northerly wind model has a larger error.

4.5 Performance of Model on Different Urban Arrangement

Figure 4.22 Illustration of Niigata urban building layout: (a) displays the building layout with black shapes representing buildings within a circular area, demonstrating their spatial structure. (b) shows the same layout but with a vertical flip.

Figure 4.22 depicts two distinct urban geometries employed in the experiment. Figure 4.22 (a) depicts the original Niigata urban geometry, which was employed to train the model for wind field prediction. Figure 4.22 (b) depicts the “new" geometry, which was generated by inverting the Niigata urban geometry. This method is employed to generate a novel geometric scene, which is then utilized to assess the model’s capacity for generalization when a new CFD simulation is not feasible.

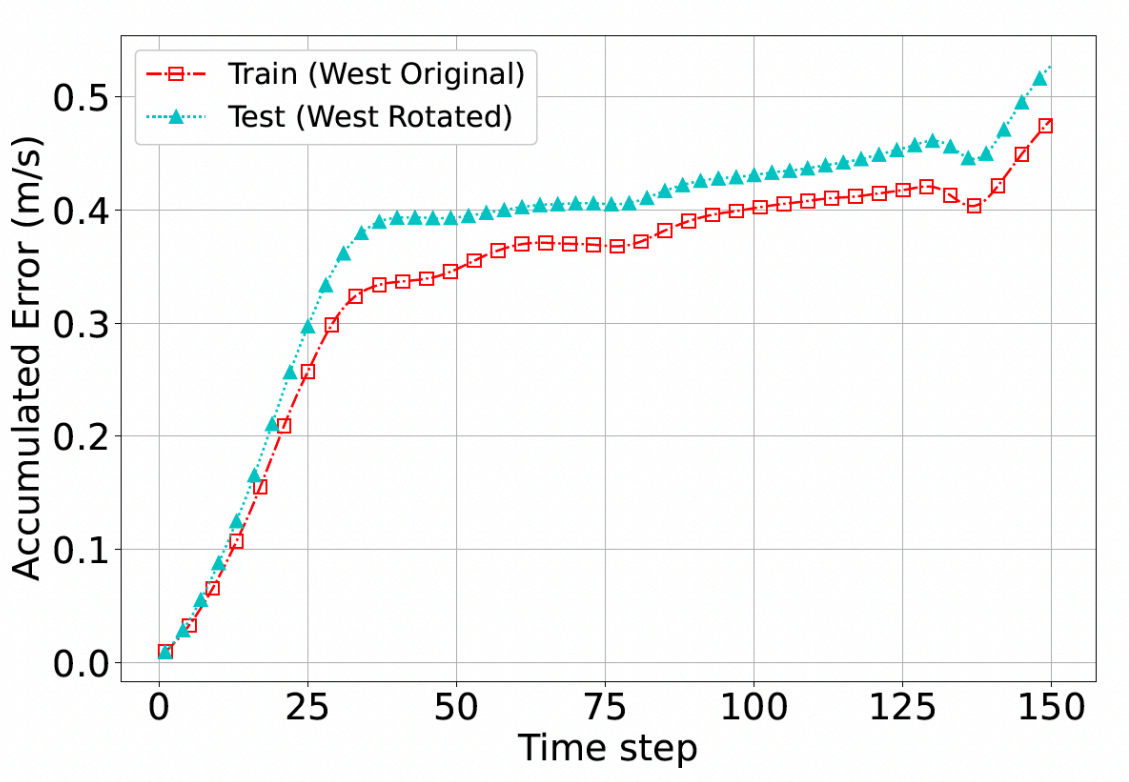

Figure 4.23 Comparison of accumulated prediction error in Niigata between the original west wind training data and the up down flipped west wind test data over 150 time steps.

Figure 4.23 illustrates the cumulative error of the FNO model trained on the West wind Niigata wind field (red line) in comparison with its generalization performance on the upside-down test set (blue triangle line). The upside-down scenario is equivalent to the generation of a novel urban geometry. It should be noted that this operation is not a strict CFD simulation. However, due to the lack of sufficient data, this method was employed to conduct generalization experiments on the wind field. As illustrated in the figure, despite the discrepancy between the upside-down test set and the original geometry, the model exhibits robust generalization capabilities. Following the flipping of the data, the error increased rapidly before T20, with the cumulative error being marginally higher than that of the original training set. However, after T50, the error stabilized and remained at approximately 0.45 m/s. This demonstrates that despite the introduction of a novel geometric configuration, the FNO model retains the capacity to discern and replicate the underlying geometric characteristics, thereby facilitating precise wind field projections.

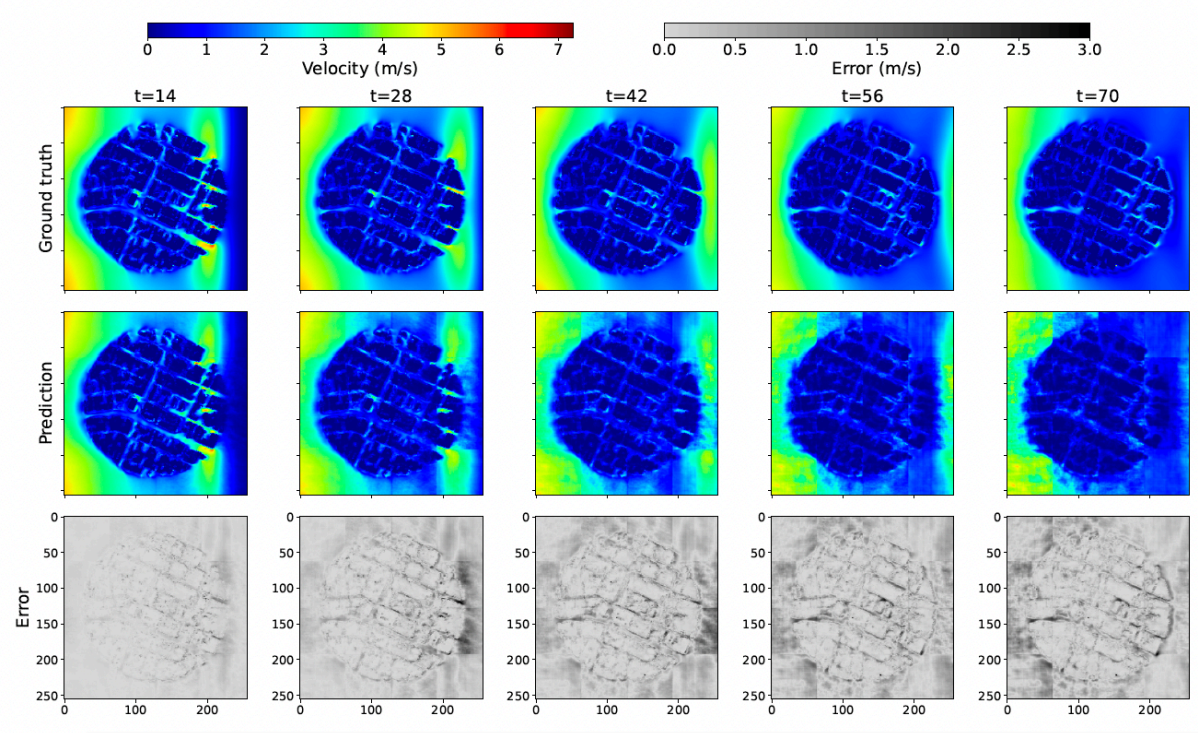

Figure 4.24 Comparison of the ground truth (top row), the predicted velocity fields (middle row), and the error distributions (bottom row) for the Niigata wind field after vertical flipping.

Figure 4.24 illustrates the outcomes of the forecast following the inversion of the Niigata west wind field. The figure depicts the true value, the predicted value, and the discrepancy between the two at various time points (t=14, t=28, t=42, t=56, t=70) for each column.

The true value (presented in the first row) depicts the authentic wind speed distribution at varying time points. The wind speed in the passageway between the buildings and the edge area exhibits a complex distribution, particularly in the flipped geometry, which displays discernible flow patterns.

The predicted value is presented in the second row. The model predicts a value that is in close proximity to the true value at the initial time steps (t=14, t=28). However, as time progresses, particularly at t=56 and t=70, the prediction error in the vicinity of the building increases. This demonstrates that although the model is capable of replicating certain global wind field characteristics following the processing of this geometric transformation, it is somewhat deficient in its ability to accurately predict long-term time steps. Error Field (Third Row): The error map illustrates the discrepancy between the predicted and true values. While the discrepancy is minimal at the initial time steps (t=14, t=28), it increases significantly at later time steps, particularly at t=56 and t=70. This is evident in the error maps, which show a larger deviation from the true value at the edge of the building and in areas with lower wind speeds. The flipped geometry presents a significant challenge to the model in long time steps, resulting

in a gradual accumulation of errors. In conclusion, the model demonstrates initial proficiency in replicating the flipped west wind Niigata wind field, yet the error rate progressively rises in extended time steps, particularly in regions with intricate geometries. These findings indicate that while the model demonstrates some capacity for generalization with regard to geometric flips, there is still scope for enhancement in the context of long-term step prediction.

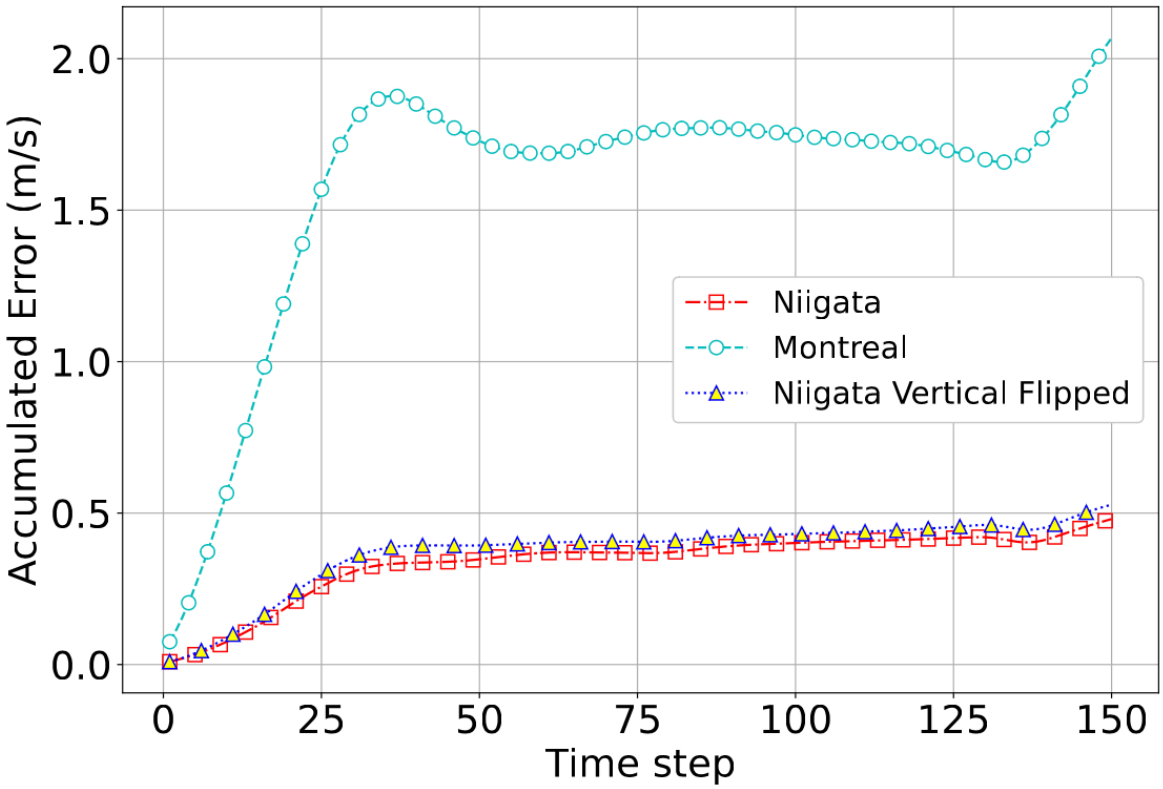

Figure 4.25 Accumulated error comparison over 150 time steps between the Niigata, Montreal, and Niigata UpDown Flipped setups.

Figure 4.25 illustrates the cumulative discrepancy in wind speed forecasts over time for three scenarios employing the Fourier Neural Operator model. The Niigata scenario, represented by the red square line, demonstrates a rapid increase in error up to approximately time step 25, after which it stabilizes. In contrast, the Montreal scenario, illustrated with a cyan circle line, demonstrates a gradual increase in error over the entire series, indicating a persistent challenge in modeling this region. The blue triangle line represents the Niigata scenario with the data flipped upside down, exhibiting a low error similar to the original Niigata data, which highlights the model's resilience to such transformations.

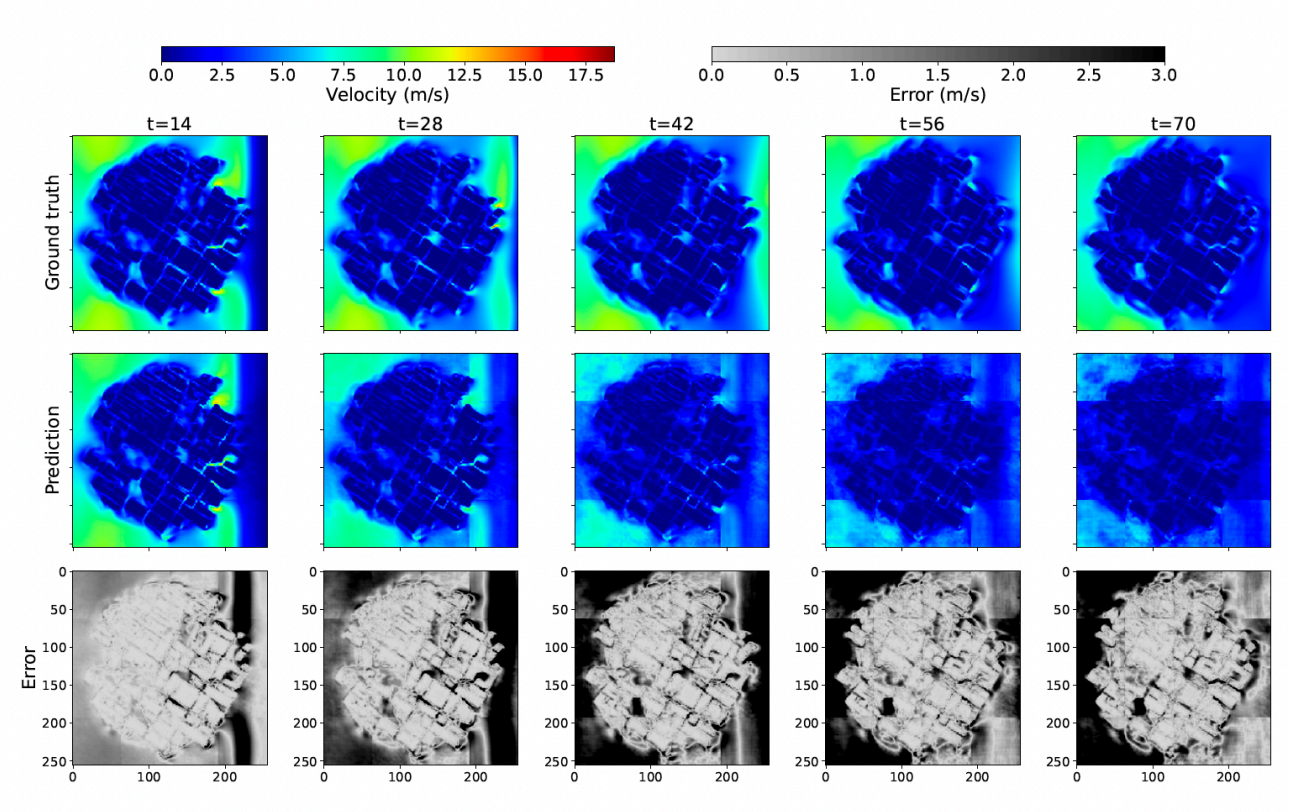

Figure 4.26 Comparison of ground truth, prediction, and error fields at time steps t = 14, t = 28, t = 42, t = 56, and t = 70 for the Montreal wind field test.

Figure 4.26 offers a comprehensive comparison of ground truth, prediction, and error fields for wind velocity predictions using the Fourier Neural Operator (FNO) at different time steps for the Montreal wind field test. The top row depicts the actual wind velocities at selected time steps (t=14, 28, 42, 56, and 70), thereby illustrating the dynamic behaviour of the wind over time. The middle row depicts the predicted velocities at the same time steps, indicating the degree of alignment between the model's predictions and the ground truth. The bottom row visualizes the absolute errors between the predictions and the ground truth, with error magnitude represented in grayscale. This format effectively highlights areas where the model performs well and where it deviates, providing valuable insights into the model's accuracy and potential avenues for improvement in urban wind field simulations.

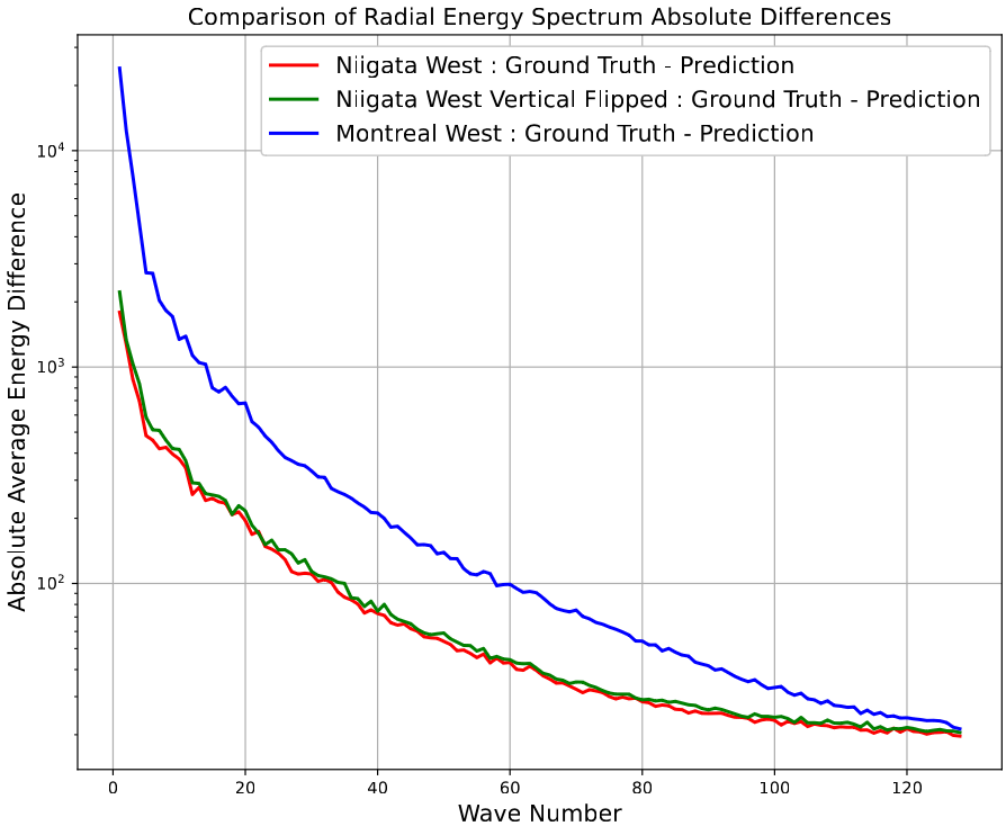

Figure 4.27 Comparison of Radial Energy Spectrum Absolute Differences between different city layout configurations.

Figure 4.27 depicts the absolute discrepancies in the radial energy spectrum between the ground truth and the predicted values for two scenarios utilizing the Fourier Neural Operator (FNO) model. The scenarios involve the wind field data from Niigata, with one set rotated upside down (red curve) and the original orientation (green curve). Furthermore, a third scenario (blue curve) presents data from Montreal's west side. The plot demonstrates that the error differences for both Niigata configurations are significantly lower and closely aligned across the wave number spectrum, indicating robust model performance even with the rotated data. In contrast, the Montreal scenario demonstrates a consistently higher error across all wave numbers, suggesting greater challenges in modeling accuracy for this region.

This comparison highlights the necessity for further model refinement to enhance prediction accuracy in complex urban settings like Montreal.

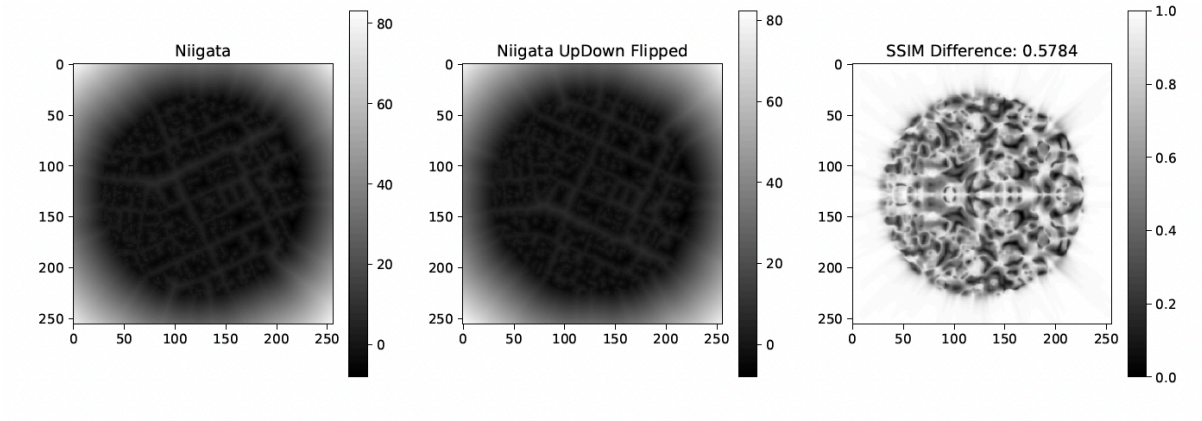

Figure 4.28 Comparison of SSIM(structural similarity) difference between Niigata original and Niigata vertically flipped wind fields.

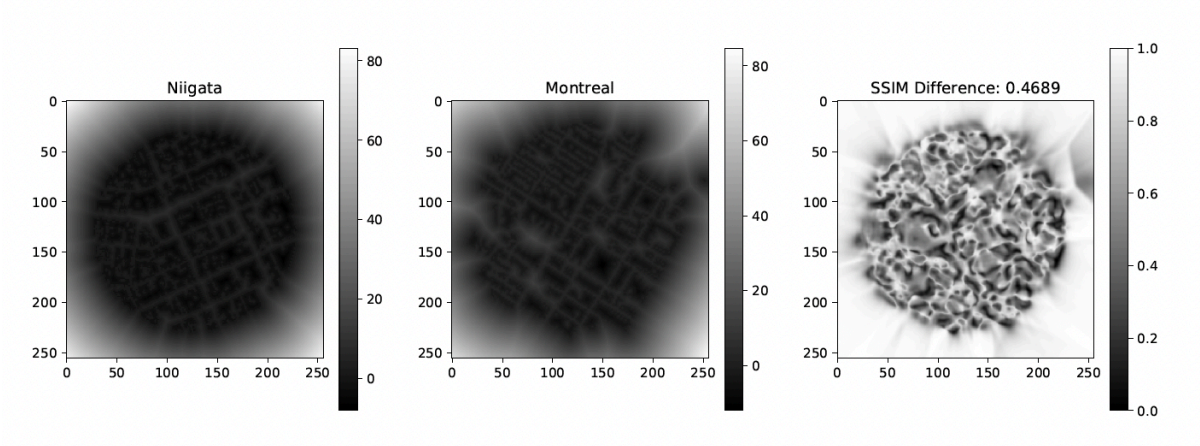

Figure 4.29 Comparison of the SSIM(structural similarity) difference between Niigata and Montreal wind fields.

Figure 4.28 illustrate a comparison of the structural similarity of the urban wind fields, assessed using the Structural Similarity Index (SSIM), for scenarios involving Niigata, both in its original form and vertically flipped, and Montreal. In Figure 4.28, the relatively small SSIM difference of 0.5784 for the original and flipped Niigata wind fields means that the structural integrity of the wind field is well preserved after the manipulation, indicating that the FNO model effectively captures and predicts the underlying dynamics even when the data orientation is changed. In stark contrast, Figure 4.29 shows a more significant SSIM of 0.4689 between the Niigata and Montreal wind fields. This larger SSIM difference underscores a significant challenge - the lower prediction accuracy of the FNO model for Montreal, where the distinctive structural and environmental characteristics differ markedly from those of Niigata. The notable decrease in SSIM values for the Montreal scenario directly correlates with a decrease in model performance, underscoring the need for improvements in the FNO model's adaptability to different urban configurations to improve forecast accuracy in diverse urban environments such as Montreal.

4.6 Time Consumption

In this subsection, we will present the time and computational resources required to train the FNO model for one iteration using the Niigata 2-meter westerly wind field data. This information is of practical significance for other researchers seeking to replicate our findings.

Table 4.1 Time Consumption Parameters

Total Training Time	82 minutes for 100 epochs
VRAM requirement	25.26 GB
Memory requirement	41 GB
Time cost for reading training data	50 seconds
A100 VRAM	32 GB

As shown in Table 4.1, the total training time for 100 epochs is 82 minutes, requiring 25.26 GB of VRAM, which is sufficient for training on hardware equipped with an Nvidia A100 GPU with 32 GB of VRAM. Additionally, loading our training data requires approximately 41 GB of memory and takes about 50 seconds.

Table 4.2 Inference Time Comparison

Method	GPU time(s)
FNO	0.006s
CityFFD	2.21s

Table 4.2 compares the GPU computation time required for the Fourier Neural Operator (FNO) and CityFFD methods. The GPU time is shown in seconds. The FNO model completes the task in 0.006 seconds, while CityFFD takes 2.21 seconds. This means that the FNO method is approximately **368 times faster** than CityFFD. This significant difference highlights the computational efficiency of the FNO model, especially in scenarios where fast computation is essential.

4.7 Ablation Study

In this section, we investigate the role of the Fourier module in accurately learning wind field features using the FNO model through ablation experiments. We conduct experiments using the complete FNO model, the FNO model with only the Fourier layers, and the FNO model without the Fourier layers. These models are trained on Niigata westerly wind data and tested on Niigata northerly wind data.

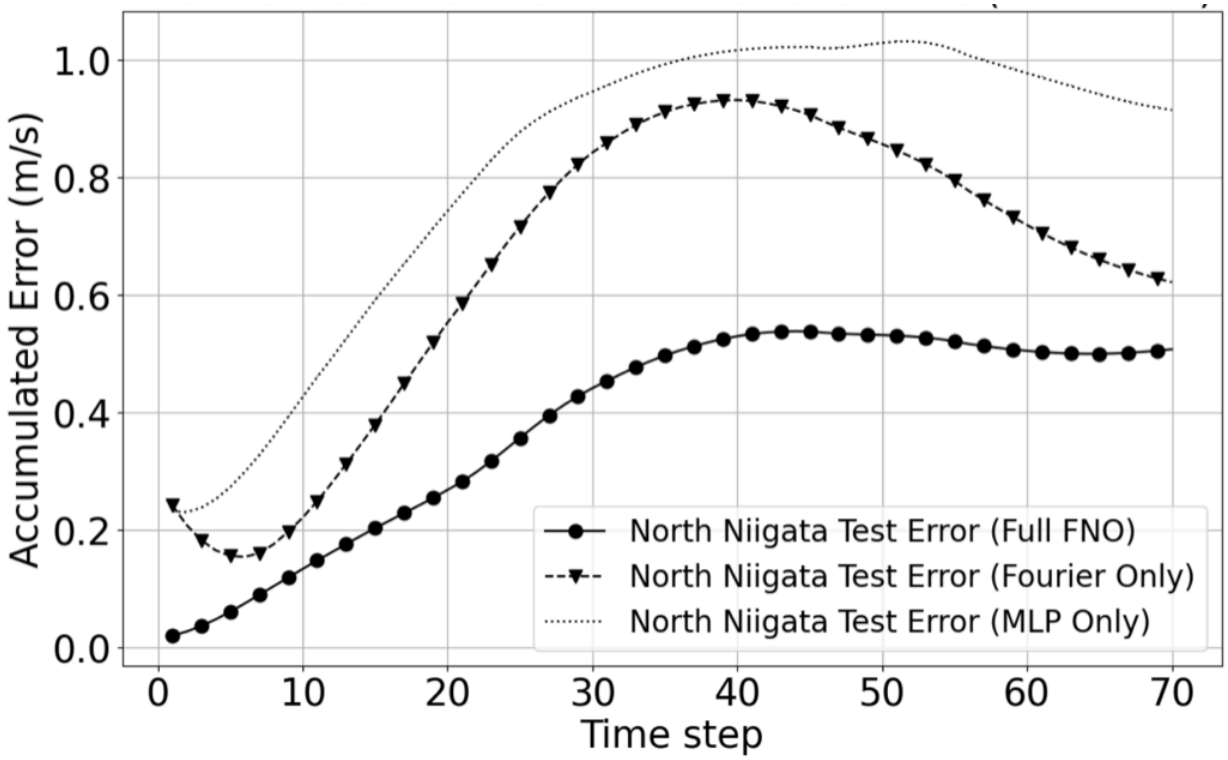

Figure 4.30 Sequential time step prediction comparison at different ablation experiment

Figure 4.30 illustrates the accumulated mean absolute error (MAE) over time for three different configurations of the FNO model on the Niigata north wind test data. Specifically, the figure compares the performance of the following three models: the full FNO model (solid black line with circles), the FNO model using only the Fourier module (black dashed line with triangles), and the FNO model using only the MLP module (black dotted line).

As shown in the figure, the full FNO model achieves the best prediction accuracy, with the lowest accumulated error across the entire time sequence, consistently maintaining a low error within the first 70 time steps. In contrast, the FNO model with only the Fourier module performs slightly worse, starting with a lower error but gradually accumulating more error as the time steps increase, eventually surpassing the full FNO model. The FNO model with only the MLP module performs the worst, with error accumulating rapidly as the time steps increase, eventually exceeding 1 meter per second.

These results indicate that the full FNO model is more effective at capturing the dynamic characteristics of the wind field, significantly outperforming the models that use only the Fourier or MLP modules. This further demonstrates the importance of the synergy between the Fourier and MLP modules in the FNO model for improving prediction accuracy.

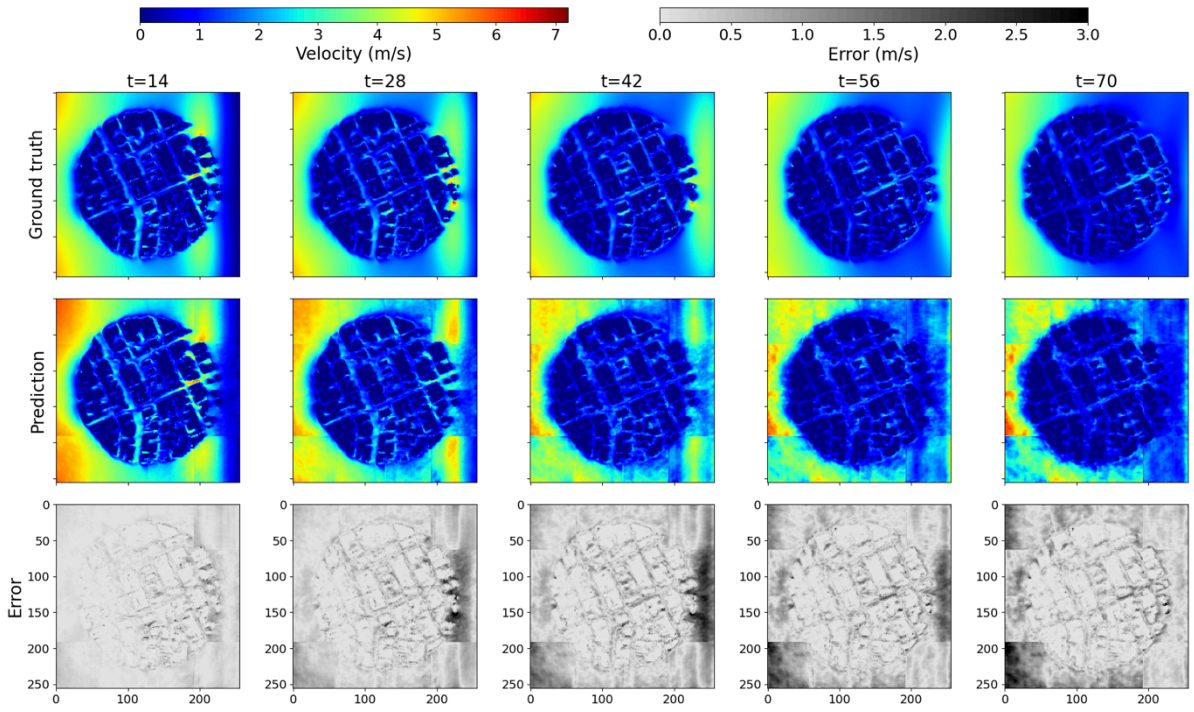

Figure 4.31 Sequential time step prediction comparison at 2m high horizontal slice, Montreal, full FNO.

Figure 4.31 presents the prediction results of the complete FNO model at different time points for the Niigata north wind field. The first row shows the ground truth of the wind field, the second row displays the predicted values from the FNO model, and the third row illustrates the difference between the ground truth and the predicted values. These images correspond to five time points: t=14, t=28, t=42, t=56, and t=70.As can be seen from the figure, in the early time steps (such as t=14 and t=28), the FNO model's predictions closely match the actual wind field, with minimal differences. However, as the time steps increase (for example, at t=56 and t=70), the discrepancies between the predictions and the ground truth gradually become more pronounced. This is particularly evident in the difference maps in the third row, especially around the edges of the wind field and near building structures.

Overall, while the FNO model is able to capture the main features of the wind field in most cases, there are still some prediction errors, especially in more complex areas or as time progresses. These errors may stem from the model's limitations in handling small-scale variations in the wind field.

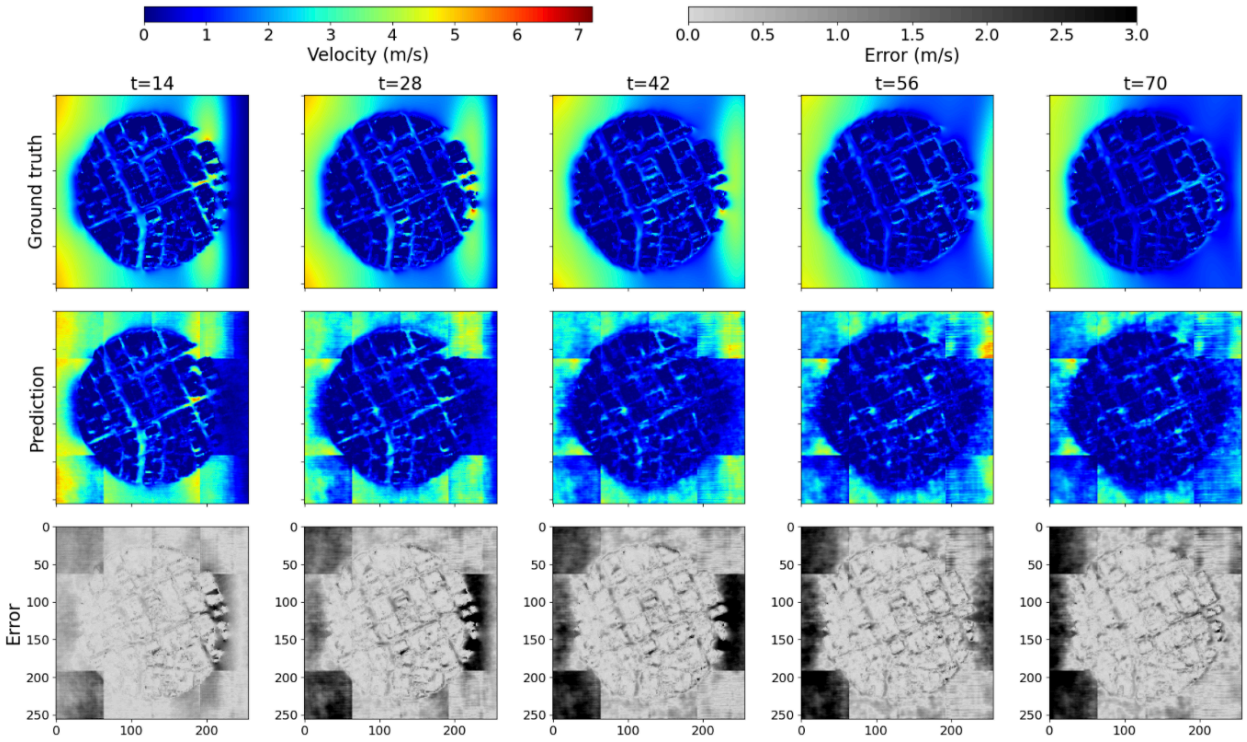

Figure 4.32 Sequential time step prediction comparison at 2m high horizontal slice, Montreal and Fourier block only.

Figure 4.32 shows the prediction results of the FNO model using only the Fourier module at different time points for the Niigata north wind field. The first row presents the ground truth, the second row displays the predictions made by the Fourier-only FNO model, and the third row shows the difference between the ground truth and the predicted values. These images correspond to five time points: t=14, t=28, t=42, t=56, and t=70. From the figure, it is evident that compared to the full FNO model, the Fourier-only FNO model performs worse in predicting the wind field. In the early time steps (such as t=14 and t=28), although the model captures some of the major features of the wind field, the discrepancies between the predictions and the ground truth are already noticeable, especially around the edges of the wind field and near buildings. As the time steps increase (such as at t=56 and t=70), the prediction errors grow more pronounced, particularly in capturing small-scale variations in the wind field.

Overall, while the Fourier module has some capability in capturing large-scale features of the wind field, its predictive accuracy significantly declines in more complex areas and as time progresses. This suggests that the FNO model using only the Fourier module is less effective in wind field prediction tasks compared to the full FNO model, and it would benefit from the inclusion of MLP modules to enhance overall performance.

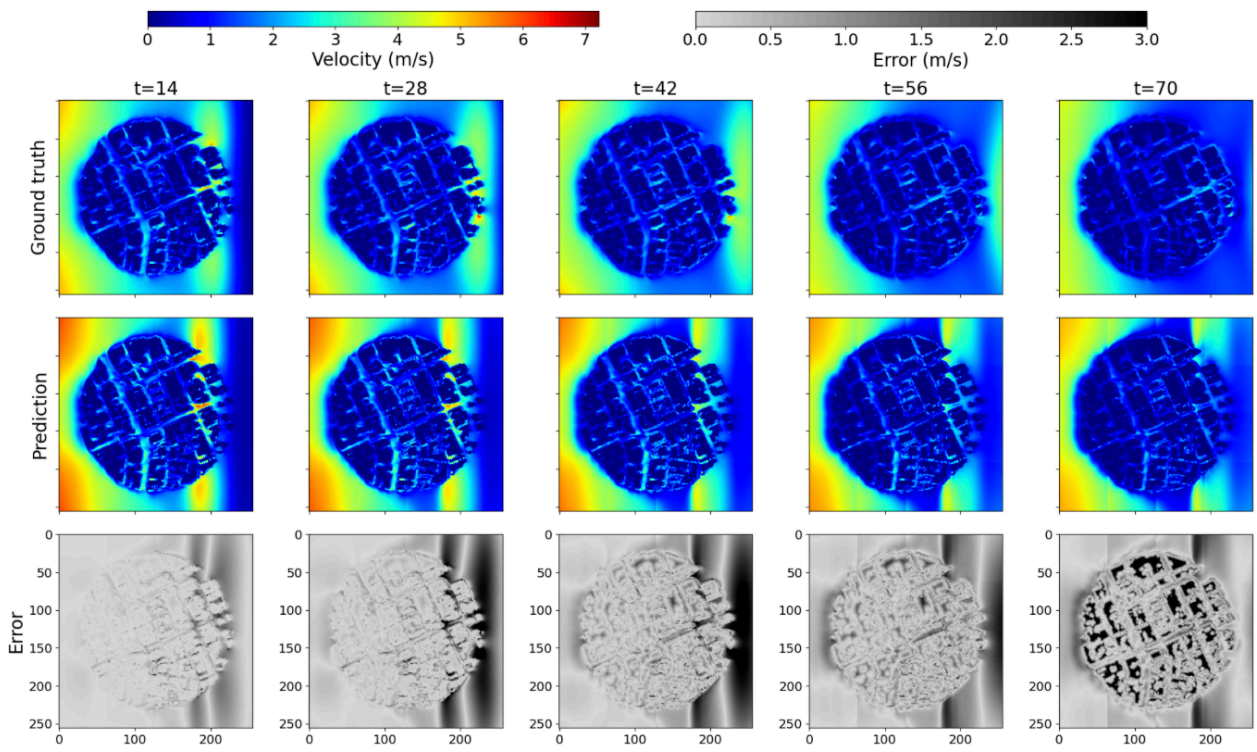

Figure 4.33 Sequential time step prediction comparison at 2m high horizontal slice, Montreal and MLP block only.

Figure 4.33 displays the prediction results of the FNO model using only the MLP module at different time points for the Niigata north wind field. The first row shows the ground truth, the second row presents the predictions made by the MLP-only FNO model, and the third row shows the difference between the ground truth and the predicted values. These images correspond to five time points: t=14, t=28, t=42, t=56, and t=70. As can be seen from the figure, compared to the full FNO model and the Fourier-only FNO model, the MLP-only FNO model performs significantly worse in predicting the wind field. Although the model captures some key features of the wind field in the early time steps (such as t=14 and t=28), its overall prediction accuracy is lower, particularly around the edges of the wind field and near buildings. As the time steps increase (such as at t=56 and t=70), the prediction errors further increase, becoming more apparent across the entire wind field, as shown in the error maps.

In summary, the MLP-only FNO model struggles to capture the complex features of the wind field, with prediction errors accumulating rapidly over time, especially when dealing with small-scale wind field variations. This indicates that the MLP-only FNO model cannot achieve satisfactory accuracy in wind field prediction tasks and would require the incorporation of the Fourier module or other techniques to improve its predictive capabilities.

Table 4.3 Ablation Study Comparison

Ablation type	Steps under 0.5m/s	Number of parameters
Original FNO	50	15187785
Fourier Block Only	19	30318705
MLP Block Only	12	15187785

Table 4.3 summarizes the results of the ablation studies on different configurations of the FNO model, specifically showing the number of time steps required to maintain an error below 0.5 meters per second and the corresponding number of parameters. Specifically:

1). **Original FNO Model**: Maintains an error below 0.5 meters per second for up to 50 time steps, with 15,187,785 parameters.

2). **Fourier Block Only**: Maintains an error below 0.5 meters per second for 19 time steps, but requires 30,318,705 parameters.

3). **MLP Block Only**: Maintains an error below 0.5 meters per second for 12 time steps, with the same number of parameters as the original FNO, in total 15,187,785.

The results indicate that the original FNO model performs the best in terms of maintaining low error over more time steps, with a moderate number of parameters. Although the Fourier-only model has more parameters, its effective time steps are significantly reduced. The MLP-only model performs the worst, with the fewest effective time steps and no reduction in the number of parameters. This suggests that the structure of the full FNO model is crucial for optimizing both the accuracy and efficiency of wind field predictions.

Chapter 5 Conclusion and Future Work

5.1 Conclusion

In this thesis, a series of in-depth learning experiments were conducted to investigate the generalization ability of the FNO model for urban wind field simulation tasks. The experiments focused on different wind directions and urban layouts, and all experiments employed the Fourier Neural Operator (FNO) model. Based on the experimental results, the following conclusions can be drawn:

(1) Due to GPU hardware limitations, the number of modes and the width that can be set in the FNO model are restricted, making training on smaller wind field blocks more effective than training on the entire wind field. Training on localized blocks allows the model to better capture dynamic features in specific areas, avoiding performance degradation caused by hardware constraints. Compared to full wind field training, block-based training improves the accuracy of localized predictions and significantly reduces computational costs. This approach is particularly efficient in resource-constrained scenarios, providing a practical solution for precise urban wind field forecasting.

(2) The absence of Signed Distance Function (SDF) data significantly degrades the model's predictive performance in complex urban wind fields, especially in densely built areas where wind field discontinuities are common. These discontinuities highlight the model's lack of physical boundary constraints, making it challenging to accurately simulate the impact of buildings and terrain. Incorporating SDF data resolves these discontinuities by providing accurate boundary information, thus enhancing the physical consistency and overall accuracy of predictions. SDF is crucial for applications such as wind energy optimization and urban pollution dispersion, where precise wind field predictions are essential.

(3) Despite being trained exclusively on west wind data, the FNO model demonstrated strong generalization capabilities when tested under north wind conditions. This suggests that the model effectively captures core dynamic features of the wind field and can adapt to untrained wind directions. The ability to generalize from west wind to north wind highlights the FNO model's flexibility, making it particularly useful in scenarios where only single-directional wind data is available for training.

(4) The FNO model's performance in cross-city applications heavily depends on the structural similarity between the target and training cities. The study showed that when the architectural layout and terrain characteristics of the target city are similar to those of the training city, such as in the flipped version of Niigata, the model maintained high prediction accuracy. However, when the target city differs significantly in structure, as in the case of Montreal, the model's predictive accuracy decreased substantially. Therefore, the structural similarity between cities is critical for successful cross-city wind field predictions, particularly in areas with dense buildings or complex terrain. Evaluating the physical characteristics of the target city against the training city is essential for ensuring accurate predictions in cross-city applications.

5.2 Future Work

The next research directions for this topic are as follows :

(1) This paper's deep learning-based FNO model for simulating micro-city wind fields has not yet made full use of the prior knowledge of CFD data. It is purely data-driven and lacks the mathematical logic support of CFD. In the next step, our work can be based on the mathematical logic foundation of CFD. By adding a physical loss function in FNO, we can make the FNO training and testing process physically interpretable.

(2) At the current stage, the FNO model and the CityFFD solver have not been fully integrated. In the next stage, the FNO model and the CityFFD solver can be designed to actually be combined to accelerate the CityFFD simulation process.

(3) Our FNO model will accumulate errors rapidly in later prediction steps after 70th step. We can reduce the error accumulation rate of the model as much as possible by increasing training data or redesigning the model structure.

(4) The experimental examples in this article are currently limited to Niigata and Montreal. Our goal is to develop an urban wind field simulation model that can be generalized in multiple cities. In the future, wind field data from more cities may be used for learning, extending the current research results to more complex urban wind field structures, and seeking to combine more advanced deep neural network technology to improve the generalization ability of the FNO model.

References

[1] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[2] Min, S., Lee, B., & Yoon, S. (2017). Deep learning in bioinformatics. Briefings in bioinformatics, 18(5), 851-869.

[3] Bakator, M., & Radosav, D. (2018). Deep learning and medical diagnosis: A review of literature. Multimodal Technologies and Interaction, 2(3), 47.

[4] Mambou, S. J., Maresova, P., Krejcar, O., Selamat, A., & Kuca, K. (2018). Breast cancer detection using infrared thermal imaging and a deep learning model. Sensors, 18(9), 2799.

[5] Zhao, X., Wu, Y., Song, G., Li, Z., Zhang, Y., & Fan, Y. (2018). A deep learning model integrating FCNNs and CRFs for brain tumor segmentation. Medical image analysis, 43, 98-111.

[6] Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. Computational intelligence and neuroscience, 2018(1), 7068349.

[7] Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. ACM computing surveys (CSUR), 54(2), 1-38.

[8] Helbing, G., & Ritter, M. (2018). Deep Learning for fault detection in wind turbines. Renewable and Sustainable Energy Reviews, 98, 189-198.

[9] Mater, A. C., & Coote, M. L. (2019). Deep learning in chemistry. Journal of chemical information and modeling, 59(6), 2545-2559.

[10] Otter, D. W., Medina, J. R., & Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. IEEE transactions on neural networks and learning systems, 32(2), 604-624.

[11] Mokayed, H., Quan, T. Z., Alkhaled, L., & Sivakumar, V. (2023). Real-time human detection and counting system using deep learning computer vision techniques. In Artificial Intelligence and Applications (Vol. 1, No. 4, pp. 221-229).

[12] Choudhary, K., DeCost, B., Chen, C., Jain, A., Tavazza, F., Cohn, R., ... & Wolverton, C. (2022). Recent advances and applications of deep learning methods in materials science. npj Computational Materials, 8(1), 59.

[13] Tyystjärvi, T., Virkkunen, I., Fridolf, P., Rosell, A., & Barsoum, Z. (2022). Automated defect detection in digital radiography of aerospace welds using deep learning. Welding in the World, 66(4), 643-671.

[14] Hao, Y., Yang, W., & Yin, K. (2023). Novel wind speed forecasting model based on a deep learning combined strategy in urban energy systems. Expert Systems with Applications, 219, 119636.

[15] Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., & Prabhat, F. (2019). Deep learning and process understanding for data-driven Earth system science. Nature, 566(7743), 195-204.

[16] Xu, X., Gao, Z., & Zhang, M. (2023). A review of simplified numerical approaches for fast urban airflow simulation. Building and Environment, 234, 110200.

[17] Carpentieri, M., & Robins, A. G. (2015). Influence of urban morphology on air flow over building arrays. Journal of Wind Engineering and Industrial Aerodynamics, 145, 61-74.

[18] Cipollina, A., Di Silvestre, M. L., Giacalone, F., Micale, G. M., Sanseverino, E. R., Sangiorgio, R., ... & Zizzo, G. (2018). A methodology for assessing the impact of salinity gradient power generation in urban contexts. Sustainable cities and society, 38, 158-173.

[19] Liu, J., & Niu, J. (2016). CFD simulation of the wind environment around an isolated high-rise building: An evaluation of SRANS, LES and DES models. Building and Environment, 96, 91-106.

[20] Toparlar, Y., Blocken, B., Maiheu, B., & van Heijst, G. J. F. (2017). A review on the CFD analysis of urban microclimate.

[21] Stam, J. (2003, March). Real-time fluid dynamics for games. In Proceedings of the game developer conference (Vol. 18, No. 11).

[22] Zuo, W., & Chen, Q. (2009). Real-time or faster-than-real-time simulation of airflow in buildings. Indoor air, 19(1), 33.

[23] Parker, J. C. (1989). Multiphase flow and transport in porous media. Reviews of Geophysics, 27(3), 311-328.

[24] F.S. Lien, E. Yee, Numerical modelling of the turbulent flow developing within and over a 3-D building array, part I: a high-resolution Reynolds-averaged Navier-Stokes approach, Boundary-Layer Meteorol. 112 (3) (2004) 427–466,

[25] J.A. Hang, Y.G. Li, Wind conditions in idealized building clusters: macroscopic simulations using a porous turbulence model, Boundary-Layer Meteorol. 136 (1) (2010) 129–159

[26] Yusuf, S. N. A., Asako, Y., Sidik, N. A. C., Mohamed, S. B., & Japar, W. M. A. A. (2020). A short review on rans turbulence models. CFD Letters, 12(11), 83-96.

[27] J.A. Hang, Y.G. Li, Wind conditions in idealized building clusters: macroscopic simulations using a porous turbulence model, Boundary-Layer Meteorol. 136 (1) (2010) 129–159

[28] R.M. Yao, Q. Luo, B.Z. Li, A simplified mathematical model for urban microclimate simulation, Build. Environ. 46 (1) (2011) 253–265

[29] J. Huang, A. Zhang, R. Peng, Evaluating the multizone model for street canyon airflow in high density cities, in: Building Simulation Conference Proceedings, 2015.

[30] Mortezazadeh, M., Wang, L. L., Albettar, M., & Yang, S. (2022). CityFFD–City fast fluid dynamics for urban microclimate simulations on graphics processing units. Urban Climate, 41, 101063.

[31] Liu, W., Sun, H., Lai, D., Xue, Y., Kabanshi, A., & Hu, S. (2022). Performance of fast fluid dynamics with a semi-Lagrangian scheme and an implicit upwind scheme in simulating indoor/outdoor airflow. Building and Environment, 207, 108477.

[32] Mortezazadeh, M., & Wang, L. L. (2020). Solving city and building microclimates by fast fluid dynamics with large timesteps and coarse meshes. Building and Environment, 179, 106955.

[33] Mortezazadeh, M., & Wang, L. (2019). An adaptive time-stepping semi-Lagrangian method for incompressible flows. Numerical Heat Transfer, Part B: Fundamentals, 75(1), 1-18.

[34] Mortezazadeh, M., & Wang, L. (2019). SLAC–a semi-Lagrangian artificial compressibility solver for steady-state incompressible flows. International Journal of Numerical Methods for Heat & Fluid Flow, 29(6), 1965-1983.

[35] Mortezazadeh, M., & Wang, L. L. (2017). A high-order backward forward sweep interpolating algorithm for semi-Lagrangian method. International Journal for Numerical Methods in Fluids, 84(10), 584-597.

[36] Katal, A., Mortezazadeh, M., & Wang, L. L. (2019). Modeling building resilience against extreme weather by integrated CityFFD and CityBEM simulations. Applied Energy, 250, 1402-1417.

[37] Mortezazadeh, M., Jandaghian, Z., & Wang, L. L. (2021). Integrating CityFFD and WRF for modeling urban microclimate under heatwaves. Sustainable Cities and Society, 66, 102670.

[38] Tominaga, Y., Yoshie, R., Mochida, A., Kataoka, H., Harimoto, K., & Nozu, T. (2005). Cross comparisons of CFD prediction for wind environment at pedestrian level around buildings. Part, 2, 2661-2670.

[39] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems, 33(12), 6999-7019.

[40] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. AI open, 1, 57-81.

[41] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019.

[42] Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

[43] Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.

[44] Weiss, Y. (2001, July). Deriving intrinsic images from image sequences. In Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001 (Vol. 2, pp. 68-75). IEEE.

[45] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics, 378, 686-707.

[46] Mano, M. (2003). Finite element method.

[47] Guo, B. (1998). Spectral methods and their applications. World Scientific.

[48] Kovachki, N. B., Lanthaler, S., & Stuart, A. M. (2024). Operator learning: Algorithms and analysis. arXiv preprint arXiv:2402.15715.

[49] Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5), 359-366.

[50] Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature machine intelligence, 3(3), 218-229.

[51] Zhou, X. S., & Huang, T. S. (2001, December). Small sample learning during multimedia retrieval using biasmap. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001 (Vol. 1, pp. I-I). IEEE.

[52] Raonic, B., Molinaro, R., De Ryck, T., Rohner, T., Bartolucci, F., Alaifari, R., ... & de Bézenac, E. (2024). Convolutional neural operators for robust and accurate learning of PDEs. Advances in Neural Information Processing Systems, 36.

[53] Liu, S. (2013, July). Fourier neural network for machine learning. In 2013 international conference on machine learning and cybernetics (Vol. 1, pp. 285-290). IEEE.

[54] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing.

[55] Kuo, F. Y., & Sloan, I. H. (2005). Lifting the curse of dimensionality. Notices of the AMS, 52(11), 1320-1328.

[56] Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., ... & Anandkumar, A. (2022). Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. arXiv preprint arXiv:2202.11214.

[57] Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., & Catanzaro, B. (2021). Adaptive fourier neural operators: Efficient token mixers for transformers. arXiv preprint arXiv:2111.13587.

[58] Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. ACM computing surveys (CSUR), 54(10s), 1-41.

[59] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., & Schmid, C. (2021). Vivit: A video vision transformer. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6836-6846).

[60] Yuan, H., Cai, Z., Zhou, H., Wang, Y., & Chen, X. (2021). Transanomaly: Video anomaly detection using video vision transformer. IEEE Access, 9, 123977-123986.

[61] Goswami, S., Bora, A., Yu, Y., & Karniadakis, G. E. (2023). Physics-informed deep neural operator networks. In Machine Learning in Modeling and Simulation: Methods and Applications (pp. 219-254). Cham: Springer International Publishing.

[62] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895.

[63] Moczo, P., Robertsson, J. O., & Eisner, L. (2007). The finite-difference time-domain method for modeling of seismic wave propagation. Advances in geophysics, 48, 421-516.

[64] Yang, S., Wang, L. L., Stathopoulos, T., & Marey, A. M. (2023). Urban microclimate and its impact on built environment–a review. Building and Environment, 238, 110334.

[65] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[66] Brook-Lawson, J., & Holz, S. (2020). CFD comparison project for wind simulation in landscape architecture. J. Digit. Landsc. Archit, 5, 318-329.

[67] Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2021). Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mechanica Sinica, 37(12), 1727-1738.