Digital Twin of PR-DNS: Accelerating Dynamical Fields with Neural Operators in Particle-Resolved Direct Numerical Simulation

Tao Zhang[1], Lingda Li[1], Vanessa López-Marrero[1], Meifeng Lin[1], Yangang Liu[1], Fan Yang[1], Kwangmin Yu[1], Mohammad Atif[1]

[1]Brookhaven National Laboratory, Upton, New York, USA

Key Points:

- The Fourier Neural Operator (FNO) model can accurately emulate particle-resolved direct numerical simulations (PR-DNS) at various resolutions.
- The computational time of the physics-based PR-DNS model is reduced by two orders of magnitude with the FNO model.
- The FNO model demonstrates robust and zero-shot generalization for various initial conditions and ultra-high resolutions.

Corresponding author: Tao Zhang, tzhang@bnl.gov

Abstract

Particle-resolved direct numerical simulations (PR-DNS) play an increasing role in investigating aerosol-cloud-turbulence interactions at the most fundamental level of processes. However, the high computational cost associated with high resolution simulations poses considerable challenges for large domain or long duration simulation using PR-DNS. To address these issues, here we present a digital twin model of the complex physics-based PR-DNS developed by use of the data-driven Fourier Neural Operator (FNO) method. The results demonstrate high accuracy at various resolutions and the digital twin model is two orders of magnitude cheaper in terms of computational demand compared to the physics-based PR-DNS model. Furthermore, the FNO digital-twin model exhibits strong generalization capabilities for different initial conditions and ultra-high-resolution without the need to retrain models. These findings highlight the potential of the FNO method as a promising tool to simulate complex fluid dynamics problems with high accuracy, computational efficiency, and generalization capabilities, enhancing our understanding of the aerosol-cloud-precipitation system.

Plain Language Summary

Particle-resolved direct numerical simulations (PR-DNS) are an important model to enhance our understanding of the fundamental processes involved in aerosol-cloud-turbulence interactions. However, achieving the extra-high-resolution simulations comes at very expensive computational cost. Although high-performance computing can accelerate PR-DNS simulations, it requires considering various factors, such as efficient MPI communications and GPU memory utilization. The machine learning digital twin models require much less computation cost compared to traditional numerical methods. Nevertheless, conventional machine learning models can only learn mappings between specific finite-dimensional spaces. The Fourier Neural Operator (FNO) method has recently been proposed to learn in the mesh-free and infinite dimensional space. In this study, we first present a digital twin model of the complex PR-DNS developed by using of the FNO method. The results show that the FNO model can achieve high accurate prediction, require low computational cost, and perform well with different initial conditions and resolutions, without re-training the machine learning models.

1 Introduction

Accurately simulating the aerosol-cloud-precipitation system and representing it appropriately in weather and climate models pose significant challenges for the cloud physics, weather, climate, and energy communities (Fan et al., 2013; Liu, 2019; Liu et al., 2023). These challenges are particularly daunting due to knowledge gaps in crucial processes that occur at scales smaller than the typical grid sizes of large eddy simulation (LES) models (e.g., 100 m). These processes include microphysics, turbulent entrainment-mixing between clouds and environmental air, turbulence, and their mutual interactions. These fundamental processes are either not represented at all or are only rudimentarily parameterized in major atmospheric models such as LES models, weather models, and climate models, thus hindering future progress in climate modeling.

Fully addressing these vital knowledge gaps at the fundamental level calls for a particle-resolved direct numerical simulation (PR-DNS) model that not only resolves the smallest turbulent eddies in clouds but also tracks motion and growth of individual particles or droplets (Gao et al., 2018). PR-DNS involves the numerical solution of the Navier-Stokes (NS) equations coupled with equations that describe the motion and growth of individual particles or droplets. The time evolution of fluid's motion as well as its thermodynamic properties (e.g., temperature and water vapor mixing ratio) is controlled by the NS equations which are solved numerically in an Eulerian framework. The motion and growth of particles/droplets, which are affected by dynamic and thermodynamic properties of surrounding fluid, are calculated in the Lagrangian framework. The hygroscopic growth of particles/droplets can subsequently affect fluid property through latent heat release and water depletion. PR-DNS is a unique tool to investigate aerosol-cloud-turbulence interactions at the process level.

Since the pioneering works in the early 2000s (P. A. Vaillancourt & Yau, 2000; P. Vaillancourt et al., 2002), a few studies have contributed to developing and applying DNS to study the interactions between cloud microphysics and turbulence. The DNS presented in (P. A. Vaillancourt & Yau, 2000) and (P. Vaillancourt et al., 2002) solves the forced incompressible Naiver-Stokes equations in 3-D by using the method of (Sullivan et al., 1994) and tracks individual droplets, to investigate the influence of turbulence on droplet size distribution under isotropic turbulent environment. Andrejczuk et al. developed a DNS model to simulate a cloud-clear air interfacial layer to investigate the effects of mixing processes on cloud microphysics in decaying moist turbulence, to examine the effects of initial turbulence kinetic energy (TKE), cloud fraction, droplet sizes, and the relationship between the mixing mechanisms and the Damköhler number(Andrejczuk et al., 2004, 2006, 2009). Kumar et al. (2013, 2012, 2014, 2017) developed a particle-resolved DNS model to study turbulent entrainment-mixing processes. Chen et al. (2016) provided the collision parameterization in a turbulent environment and then investigates the impact of turbulence on collision efficiency and the droplet size distribution in cumulus clouds (Chen, Yau, & Bartello, 2018). Additionally, subsequent studies by Chen, Yau, Bartello, and Xue (2018) and Chen et al. (2020) investigate the effect of aerosols and turbulence on cloud droplet growth.

However, PR-DNS requires a large amount of computational resources, which limits its current use to relatively small-domain simulations (Gao et al., 2018). The parallel algorithm is designed by decomposing the domain into subdomains. The computation of a subdomain is conducted on an independent processor, and the required exchange of boundary information between the processors is done through Message Passing interface (MPI). Due to the communication bottleneck, the linear scaling breaks down rapidly as the number of CPU cores is further increased, which significantly inhibits our ability to scale up the PR-DNS simulations. Linear solvers such as the Portable, Extensible Toolkit for Scientific Computation (PETSc, Balay et al. (2019)) and High-Performance Preconditioners (HYPRE, Falgout and Yang (2002)) are used in the implementation of the Navier-Stokes equations in our original PR-DNS (Gao et al., 2018). PETSc is a flex-

ible and highly efficient framework for solving large-scale numerical problems, including parallel solvers for PDEs. HYPRE is a library of scalable preconditioners for solving large-scale PDEs. However, high efficiency using PETSc and HYPRE requires tuning their parameters, such as the pc_hypre_boomeramg_strong_threshold value when using the Hypre BoomerAMG preconditioner, which could impact the convergence rate and efficiency of the solver. Van Heerwaarden et al. (2017) proposed GPU-based DNS with good speedup on a single GPU. However, a major limitation in single-GPU computation is its available memory capacity, which poses a hard constraint on the maximum size of computational mesh. In addition, traditional solvers impose a trade-off on the resolution: coarse grids are fast but less accurate; fine grids are accurate but slow. Complex systems of partial differential equations (PDE) usually require a very fine discretization, and are therefore very challenging and time-consuming for traditional solvers.

Recently, a number of studies have attempted the use of deep neural networks for simulating turbulent fluid flows (Raissi et al., 2019; Bhatnagar et al., 2019; Xie et al., 2019; Kochkov et al., 2021). The main advantage of neural network-based solvers is that upon training they incur significantly lower computational overhead compared with traditional numerical solvers. However, neural network-based solvers, like video frame prediction, can only learn mappings between specific finite-dimensional spaces. As a result, if the resolutions change, these models have to be retrained, which could limit their practical applications. As a solution, the resolution-invariant neural networks are needed to allow for greater flexibility and adaptability for super-resolution modeling.

Recently, a new line of work proposed learning mesh-free, infinite dimensional operators with neural networks (Lu et al., 2021; Li et al., 2020b). These approaches, so called "neural operators", only need to be trained once and have the ability to transfer solutions to different mesh granularities. They have shown promise in improving the accuracy and efficiency of numerical simulations (Li et al., 2020a; Lu et al., 2021). Nevertheless, the neural operators currently in use incorporate integral operators, which can be computationally demanding and time-consuming. As such, there is a need for more efficient computational optimization to improve the scalability and practicality of these methods. The Fourier Neural Operator (FNO) method has recently been proposed to learn a continuous function via representing the model in its function space (Li et al., 2020a). The integral operator is restricted to a convolution, and instantiated through a linear transformation in the Fourier domain with high computational efficiency.

As the first application of the FNO to the PR-DNS model, here we build several digital twin surrogate models of our PR-DNS and use numerical simulation results to evaluate their accuracy and computational performance. The digital-twin PR-DNS models can be used to directly simulate turbulent flows on a larger model domain, covering more realistic scales in turbulent clouds. Particularly, the FNO surrogate model demonstrates better accuracy than other deep learning-based methods, such as UNET(Ronneberger et al., 2015) and ResNet(He et al., 2016) in terms of the prediction accuracy for velocity and temperature. Its computational cost is two orders of magnitude lower than the original PR-DNS model, especially for high resolution. In addition, it exhibits good generalization ability for different initial conditions and zero-shot super resolution, which do not require retraining the deep learning model, instead, only utilizing the pre-trained model for high resolution.

The rest of the paper is organized as follows: Section 2 introduces the PR-DNS model and the simulation data. Section 3 describes the FNO algorithm. Section 4 discusses the performance of FNO method, including comparison with previous studies. Conclusions and discussions are given in Section 5.

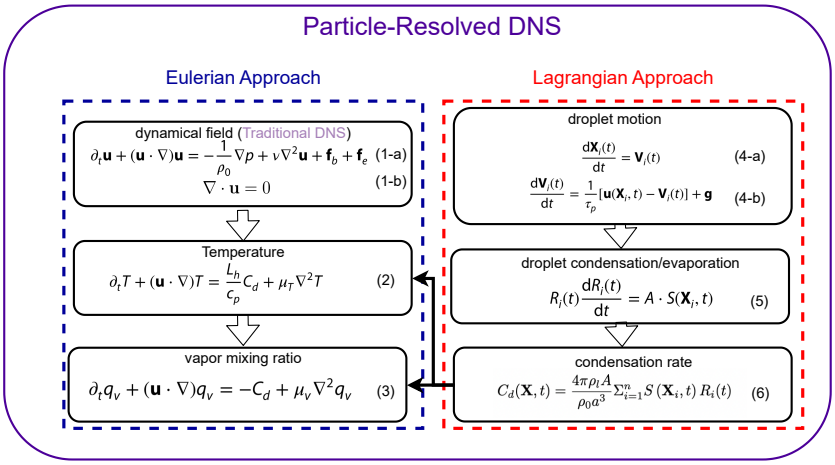

Figure 1. PR-DNS overview

2 Model and Data

We use the PR-DNS developed by Gao et al. (2018), which includes the Eulerian field representation of turbulence near the Kolmogorov microscale and Lagrangian droplet tracking, shown in Figure 1. The dynamical model is based on the incompressible Boussinesq fluid system, given by Equations 1-a and 1-b, Figure 1. The $\mathbf{u}$ is the velocity field, p is the pressure field, ν is the kinematic viscosity, and ρ_0 is the density of dry air. $\mathbf{f}_b$ and $\mathbf{f}_e$ are the buoyancy and large-scale forcings, respectively. The buoyancy force is given by

$$\mathbf{f}_b = -\mathbf{g}\left[\frac{T - T_0}{T_0} + 0.608(q_v - q_{v0}) - q_c\right] \tag{7}$$

which depends on the gravity g, the temperature T, and vapor mixing ratio q_v. It thus couples the fluid velocity equations with T, q_v and thus cloud microphysics. The external force $\mathbf{f_e}$ maintains a statistically stationary homogeneous turbulence.

In the equations for temperature (Eq 3) and vapor mixing ratio (Eq 4) from Figure 1, L_h is the latent heat of water vapor condensation, c_p is the specific heat, and μ_T and μ_v are, respectively, the molecular diffusivity for temperature and water vapor. The condensation rate C_d (Eq 6) from Figure 1 depends on the droplet radii $R_i(t)$ and thus couples the Lagrangian particles and the Eulerian field.

For the Lagrangian approach, equations 4-5 from Figure 1 describe the motion and condensation/ evaporation of cloud droplets, where $R_i(t)$, $X_i(t)$, and $V_i(t)$ denote, respectively, the radius, position coordinate, and velocity of the i-th droplet at time t.

In our simulations, the physical solution domain is set to be $0.512 \times 0.512\ m^2$ in the two-dimension space with double periodic boundary conditions. In order to evaluate the performance at different resolutions, we conduct the PR-DNS simulations with 64 x 64, 128 x 128 and 256 x 256 meshes which correspond, respectively, to 8mm, 4mm and 2mm grid sizes. In addition, in order to evaluate the performance of the deep learning models given different initial conditions, we conduct the PR-DNS with different initial conditions for the 128x128 grid. The time step size is 0.003s on average, satisfying the Courant-Friedrichs-Lewy condition. We run each simulation for 6000 time steps. The first 4000 time steps are used to train the deep learning methods, with the rest of 2000 time steps for performance validation.

3 Method

3.1 Fourier Neural Operator

The traditional PDE solvers, such as finite volume and finite difference, typically start with the initial conditions and then iterate forward to generate the solution at each time step based on the solution at the previous time steps. For the original PR-DNS model, given the specific initial conditions, it outputs the properties at each time step, such as velocity, temperature and water vapor. In other words, the PDE solvers build the mapping $\mathcal{G}_\theta$ from the initial condition $\mathcal{A}$ to the solution $\mathcal{U}$, $\mathcal{G}_\theta$: $\mathcal{A} \times \theta \to \mathcal{U}$. Deep learning models have been proved to be excellent function approximators in many domains, and they can emulate the initial-condition-to-solution mapping of PR-DNS too.

Two distinct types of deep learning methods have the potential as the digital twins of the traditional solvers, as illustrated in Figure 2. Because physical variable distributions in a discretized 2D/3D space resemble 2D/3D images, the first category leverages existing deep learning techniques developed for computer vision, such as UNet and Residual neural network (ResNet). UNet is a well-known deep neural network architecture that is widely employed for image segmentation in computer vision, utilizing the encoder-decoder structure to reduce the feature dimension (Ronneberger et al., 2015). The encoder downsamples the input by a series of convolutional layers, while the decoder upsamples the feature to produce the final output. In addition, the decoder uses the skip connections to embed the feature learned by the encoder that preserve the spatial information of the original input. In contrast, ResNet addresses the issue of vanishing gradients in very deep networks by using residual blocks that enable gradients to propagate more easily through the network (He et al., 2016). In contrast to the UNet architecture, the ResNet replace the convolutional layers by the residual blocks. Both UNet and ResNet can function as PDE solvers by mapping one finite space to another finite space, as depicted in Figure 2(a). However, when the resolution or shape of the application domain change, the deep learning models have to be re-trained. Conversely, neural operators are specially designed to learn mathematical operators that are independent to the discretization, such as differential operators or integral operators, instead of directly learning a functional mapping that binds to a specific discretization. Figure 2(b) illustrates that the neural operator attempts to learn the integral operators in the Green Function, which can find analytical solutions to PDE equations.

Li et al. (2020b, 2020a) proposed the neural operator that learns the mapping $\mathcal{G}_\theta$:

$$\mathcal{G}_\theta = Q\left((K_l + W_l) \circ \sigma_l \circ \cdots \circ (K_0 + W_0) \circ \sigma_0\right) P \tag{8}$$

This neural operator includes the P and Q transformation network, integral operator K, linear operators W and the activation function σ for non-linear mapping, as illustrated in Figure 3. P can be interpreted as the encoder which employs neural networks to map inputs into a high dimension latent space, and Q serves as the decoder that projects outputs back to the original space. Both P and Q are shallow fully-connected neural networks. The integral operator K is defined as:

$$\left(\mathcal{K}(a;\phi)v_t\right)(x) := \int_D \kappa(x, y, a(x), a(y); \phi) v_t(y) \mathrm{d}y, \quad \forall x \in D \tag{9}$$

where $a \in \mathcal{A}$ denotes the input, ϕ represents the learnable parameters in neural networks, v_t denotes the $\sigma_t(K_t + W_t)$ in the equation 8. κ refers to the periodic spatial kernel function, which is in accordance with the PR-DNS periodic boundary. The integral operator $\mathcal{K}$ is time consuming. To address this challenge, Li et al. (2020a) proposed the Fourier Neural Operator (FNO), which takes advantage of the Fast Fourier Transform (FFT) algorithm to calculate the calculation of the integration.

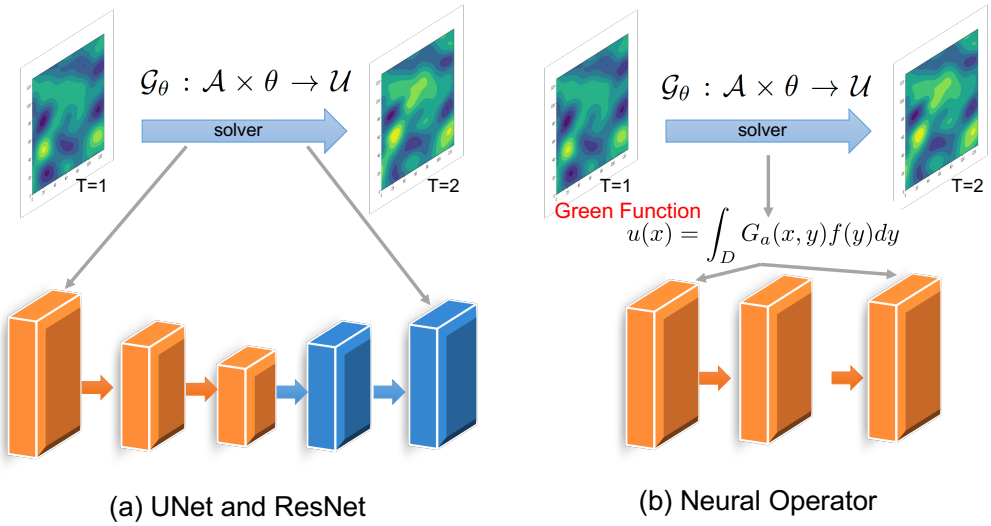

Figure 2. Two kinds of deep learning PDE solvers, (a) finite mapping based on traditional deep learning, such as UNet and ResNet; (b) infinite mapping based on neural operator.

$$\left(\mathcal{K}(a;\phi)v_t\right)(x) = \mathcal{F}^{-1}\left(\mathcal{F}\left(\kappa_\phi\right) \cdot \mathcal{F}\left(v_t\right)\right)(x) \tag{10}$$

where $\mathcal{F}$ denotes the Fourier transform of a function f and $\mathcal{F}^{-1}$ is its inverse:

$$(\mathcal{F}f)_j(k) = \int_D f_j(x) e^{-2i\pi\langle x,k\rangle}\,\mathrm{d}x \tag{11}$$

$$\left(\mathcal{F}^{-1}f\right)_j(x) = \int_D f_j(k) e^{2i\pi\langle x,k\rangle}\mathrm{d}k \tag{12}$$

3.2 Hyperparameters and Training

The FNO architecture is generically depicted in Figure 3. In the present study, the FNO architecture comprises six layers in total, including two shallow fully-connected neural networks (P and Q) and four Fourier layers. P has 32 output channels, while Q has 128 output channels. In each Fourier layer, under the assumption that κ is periodic, it can be represented by a Fourier series expansion. To effectively capture the relevant components, each Fourier layer only retains the lowest k Fourier modes to learn the low-frequency components in PDE. In this study, we found that 30 modes can achieve the best performance. For the activation function, we employ the Gaussian Error Linear Unit (GELU), which is a smoother version of the Rectified Linear Unit (ReLU).

The deep learning models were trained on a single NVIDIA GeForce RTX 3090 with 24GB memory using the PyTorch (Paszke et al., 2019) framework. Each model was trained for 500 epochs with a batch size of 50. We utilized the Adam optimizer, a first-order gradient descent method, with a weight decay of 10^{-4}. The initial learning rate was set to 0.001 and was halved every 100 epochs. For training and testing, we used the L2 based loss function, as defined in Eq. 13.

$$\zeta_2 = \frac{\sqrt{\Sigma_{i=1}^{m}(u(x_i) - \hat{u}(x_i))^2}}{\sqrt{\Sigma_{i=1}^{m}(u(x_i))^2}}, \tag{13}$$

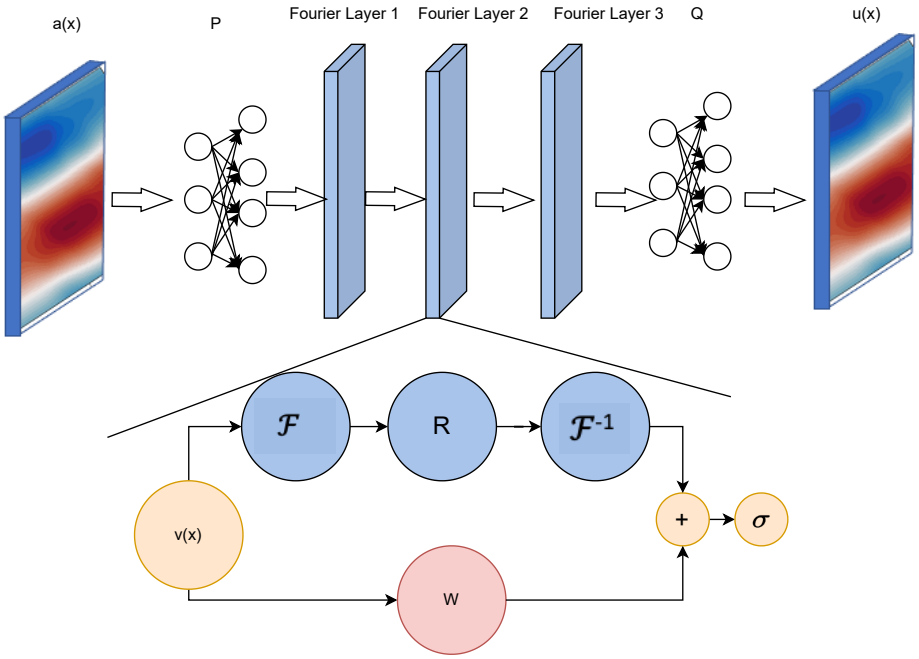

Figure 3. FNO framework. The input consists of 10 previous time steps of one variable, such as velocity or temperature. The output is variable at the 11th time step. The FNO framework consists of two shallow fully-connected neural networks, and four Fourier layers.

where m is the total number of grids, u is the ground truth and $\hat{u}$ is the prediction by deep learning models.

4 Results

In this section, we compare FNO models with ResNet and UNet in their ability to emulate the fields of air velocity and temperature. In the UNet architecture used in this study, the encoder consists of three convolutional blocks with 16, 32, and 64 output channels, respectively, while the corresponding decoder has 64, 32, and 16 output channels. The ResNet has a similar U-shaped structure, but replaces vanilla convolutional blocks used in UNet with residual blocks. For each of the three deep learning models, we used three resolutions, 64 x 64, 128 x 128 and 256 x 256. For fairness, all methods use the same training and testing datasets for each resolution. For the 64 x 64 test case, we used N=2000 training time steps and 1000 testing time steps. For the 128 x 128 and 256 x 256 test cases, we used N=4000 training time steps and 2000 testing time steps.

4.1 Accuracy

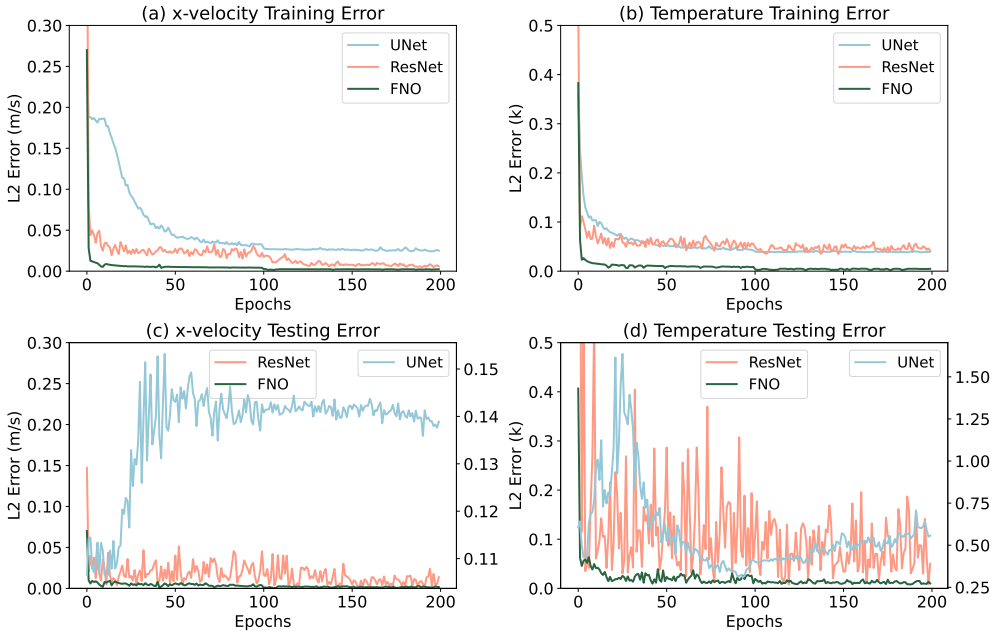

Figure 4. Training/testing errors as a function of epochs for UNet, ResNet, and FNO for the x component of velocity (a,c) and temperature (b,d) at the 64x64 resolution.

Figure 4 shows the training and testing errors as a function of the number of epochs. It reveals that FNO exhibits the lowest training and testing errors when compared to ResNet and UNet, not only at the end of but also throughout the training procedure. Furthermore, the FNO model converges faster. Remarkably, in the case of the x-velocity variable, the UNet exhibits poor performance in both training and testing. Similarly, for the temperature variable, ResNet exhibits unstable and fluctuating testing performance.

Table 1 provides additional evidence that FNO performs well across various resolutions, except the temperature at the R-256 resolution. Notably, the testing error for the x-velocity variable remains consistent across different resolutions, indicating that the performance is resolution invariant.

Fig 5 and Fig 6 display 2D spatial maps of the x-velocity and temperature variables at the 128x128 resolution, respectively, at a single time snapshot as examples. In the case of the x-velocity, the UNet method can capture the overall pattern of high ve-

Table 1. Testing L2 errors for the various resolutions, including 64x64 (R-64), 128x128 (R-128), and 256x256 (R-256).

	x-velocity			temperature		
	R-64	R-128	R-256	R-64	R-128	R-256
UNet	0.997	0.058	0.078	0.722	1.036	3.603
ResNet	0.001	0.0009	0.003	0.024	0.391	0.043
FNO	0.0001	0.0001	0.0007	0.008	0.013	0.196

locity at the bottom and low velocity on top. However, the contour details are not accurate. Conversely, the ResNet and FNO methods closely resemble the ground truth obtained from PR-DNS. With regard to the temperature variable, the UNet method tends to produce slightly larger predictions than the ground truth. The ResNet method achieves slightly higher values than the ground truth in the lower part of the map. Meanwhile, the FNO method captures the temperature pattern most accurately.

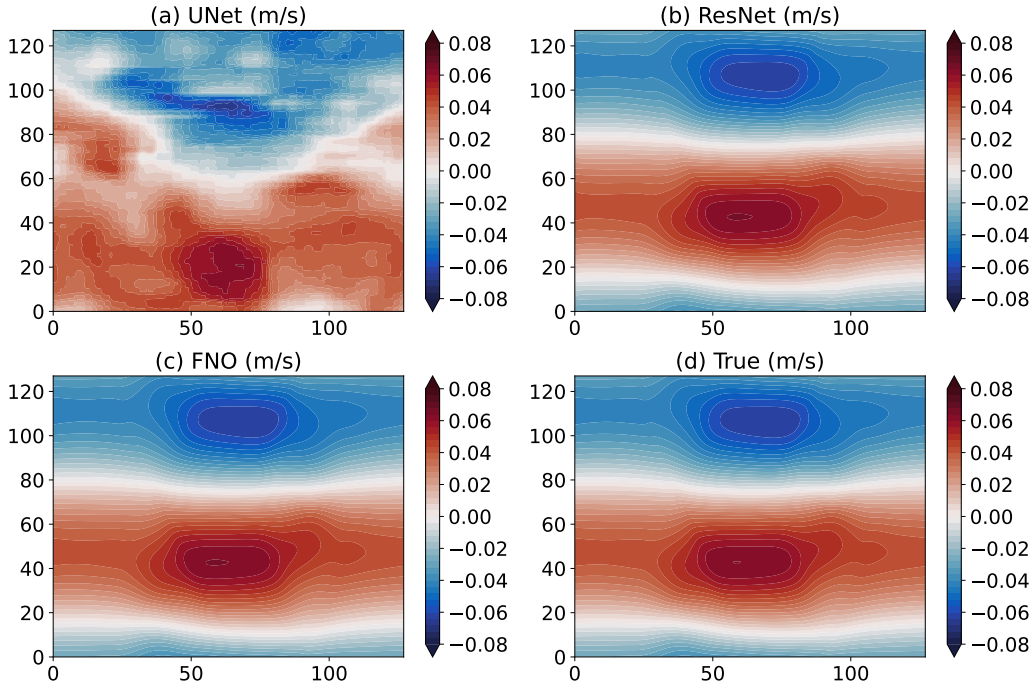

Figure 5. Snapshot prediction of x-velocity by UNet (a), ResNet (b), FNO (c), ground truth (d)

In addition to the single time snapshot, we show in Figures 7 and 8 the L2 error between deep learning predictions and ground truth for the entire testing dataset on a grid point by grid point basis, in order to quantitatively compare their accuracy. The calculation of this error is performed as follows:

$$\zeta(x_i, x_j) = \frac{\sqrt{\Sigma_{k=1}^{T}(\hat{u}(x_i, x_j) - u(x_i, x_j))^2}}{\sqrt{\Sigma_{k=T}^{t} u(x_i, x_j)^2}}, \tag{14}$$

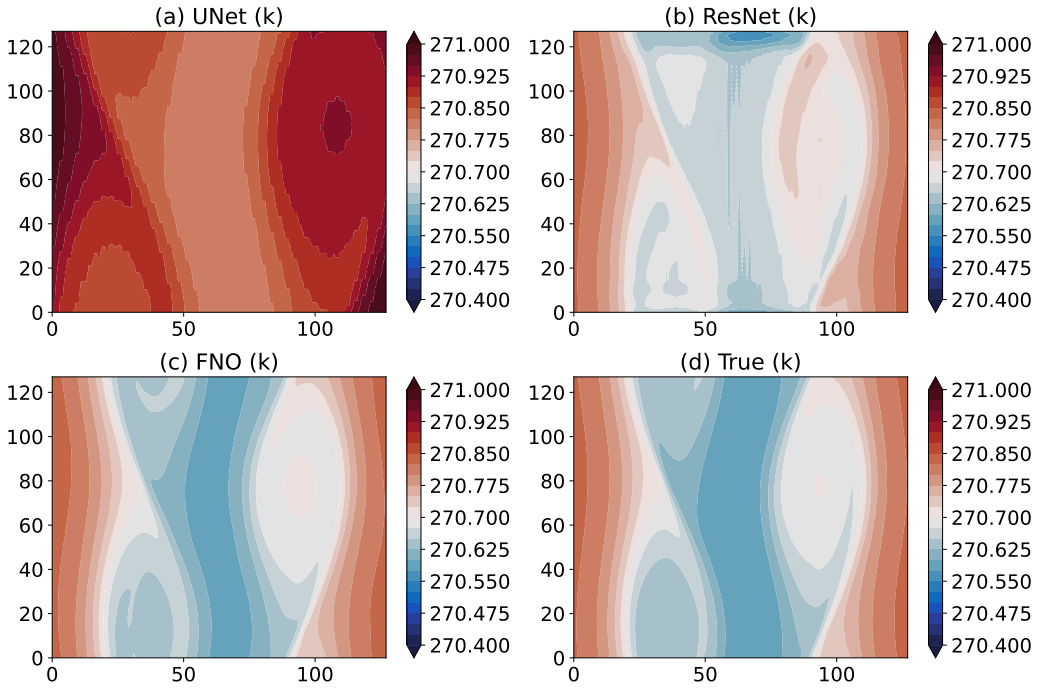

Figure 6. Same as Figure 5, but for temperature

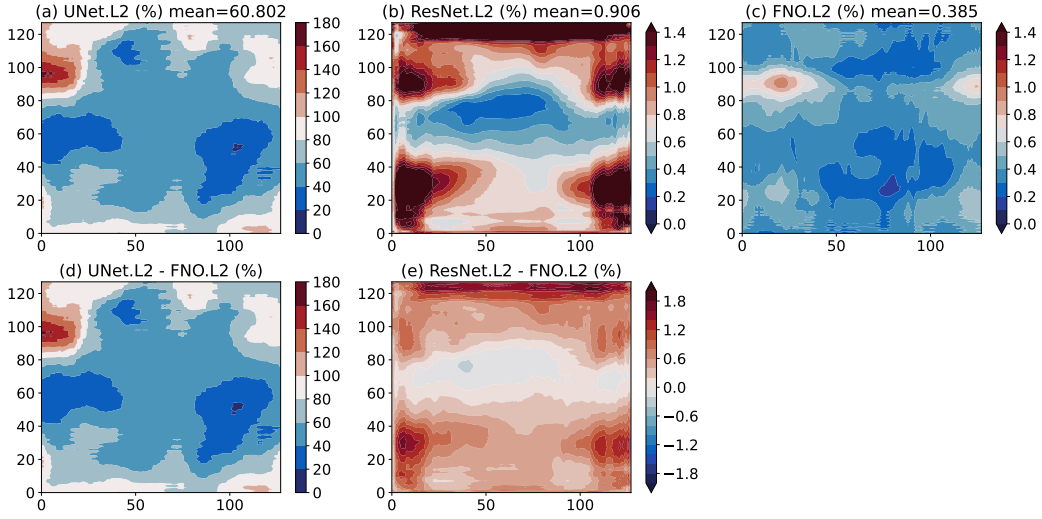

Figure 7. Snapshot prediction accuracy of x-velocity at every grid point by UNet, ResNet, and FNO

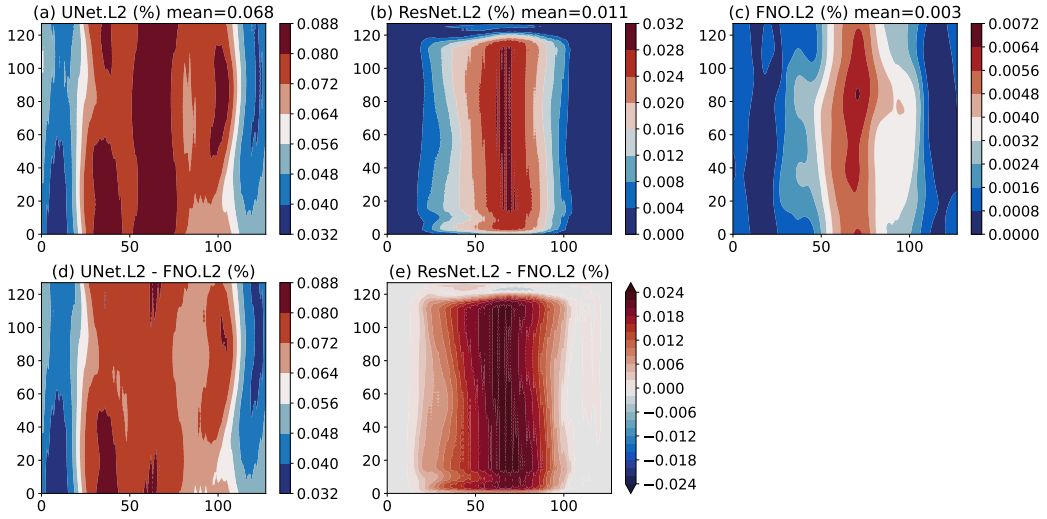

Figure 8. Same as Figure 7, but for temperature

where i and j represent the spatial coordinates in the 2D space. T is the total length of the testing dataset, $\hat{u}$ is the prediction by deep learning models, and u denotes the corresponding truth.

For the x-velocity variable, the L2 error of the UNet method is exceedingly high at 60%. In contrast, the ResNet and FNO methods achieve significantly lower L2 errors of 0.905% and 0.385%, respectively. Notably, the L2 error of ResNet is slightly larger than that of FNO. With regard to the temperature variable, although the UNet method decreases the error, it remains the worst among all three methods. Conversely, the FNO method achieves the lowest error averaged across all grid points.

4.2 Computational efficiency

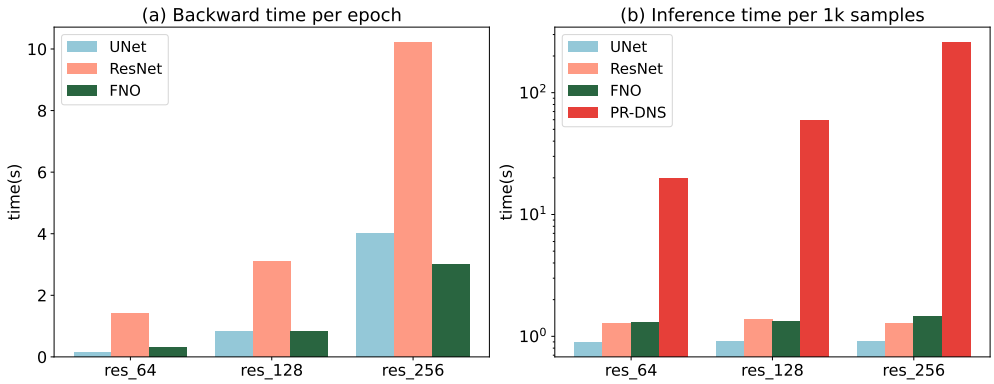

Figure 9. Computational cost for (a) backward process per epoch, (b) inference per 1k samples and 1k time steps in the original PR-DNS model under different resolutions.

Figure 9 evaluates the computational efficiency of UNet, ResNet, FNO, and the numerical PR-DNS solver across multiple resolutions. The deep learning models consist of

both forward (inference) and backward steps. For the inference step, input data is fed into the deep learning models and then the models compute the output predictions through the neural network. The backward step computes the gradients of the loss function with respect to the parameters of the neural network and updates the parameters correspondingly to minimize the loss function. During training, both forward and backward steps are invoked, while only the forward step is required at the prediction time.

The results show that all three deep learning methods are significantly less computationally expensive, by two orders of magnitude, compared to the numerical PR-DNS solver. We observe that while the inference time of the numerical PR-DNS model increases roughly linearly with the resolution, those of ML models only increase slightly, demonstrating their efficiency for large domains. It is worth noting that the inference time of FNO is slightly larger than that of UNet and ResNet. The reason could be that although FNO has fewer layers (8) compared to UNet (13) and ResNet (17), it involves a complex Fourier transforming process in the Fourier layer, which takes 3 times longer than the 2D convolution operations. The backward time of ResNet is the largest due to its more layers and parameters, while FNO has the smallest backward time due to its fewer layers and parameters.

4.3 Generalizability

Traditional PDE solvers typically require re-running when the initial condition or resolution changes. Fortunately, due to the generalizability of deep learning models, deep learning methods provide the advantage of zero-shot learning, meaning that pre-trained models can perform well on new data without the need for additional training. It is worth noting that all three deep learning models used in the present study, namely FNO, ResNet, and UNet, can achieve zero-shot learning under different initial conditions, but only FNO can do so under different resolutions. In this section, we only focus on FNO because, as discussed in Section 4.1, the ResNet has similar accuracy to FNO and UNet can not achieve satisfactory performance.

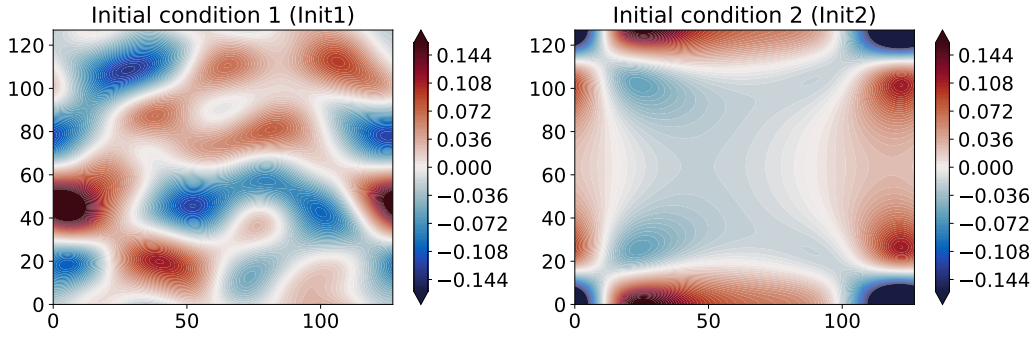

Figure 10. Two initial conditions for x-velocity.

We conduct two PR-DNS simulations, each starting with different initial conditions for the x-velocity. These two initial conditions are shown in Figure 10. Each PR-DNS simulation has the same initial condition for the temperature field. However, due to the coupling of the equations in the PR-DNS model (refer to Section 2 and Figure 1) and the fact that the two simulations have different initial conditions for the x-velocity, the time evolution of the temperature field will be different in each case. For reference, the PR-DNS simulation snapshots, that is, our ground truth, are shown in Figures 11 (d) and (e) for x-velocity and in Figures 12 (d) and (e), for temperature.

The deep learning model, FNO, can be trained and tested independently for each simulation, as shown in Figures 11-12 (a,d) and Figures 11-12 (b,e), respectively. As demonstrated previously via Figures 5-6, FNO can achieve high accuracy in this scenario. For the cases now under discussion, the normalized L2 errors between the ground truth and FNO prediction for x-velocity are 0.0001 when training and predicting using data from the first PR-DNS simulation only and 0.0003 when training and predicting with data from the second PR-DNS simulation only. In the case of temperature, the corresponding normalized L2 errors are 0.013 and 0.007.

To test the generalization capability with respect to the initial conditions, we train the model using data from the first PR-DNS simulation, labeled "init1" in Figures 10-11, and predict using data from the second PR-DNS simulation, labeled "init2" in Figures 10-11. This means that we do not need to re-train the deep learning model under other initial conditions. The FNO predictions under this scenario are shown in Figure 11 (c) for x-velocity and Figure 12 (c) for temperature, where one can observe that the zero-shot learning for different initial conditions can achieve high accuracy as well. The L2 errors of the FNO predictions under this zero-shot learning for x-velocity and temperature are, respectively, 0.0001 and 0.094.

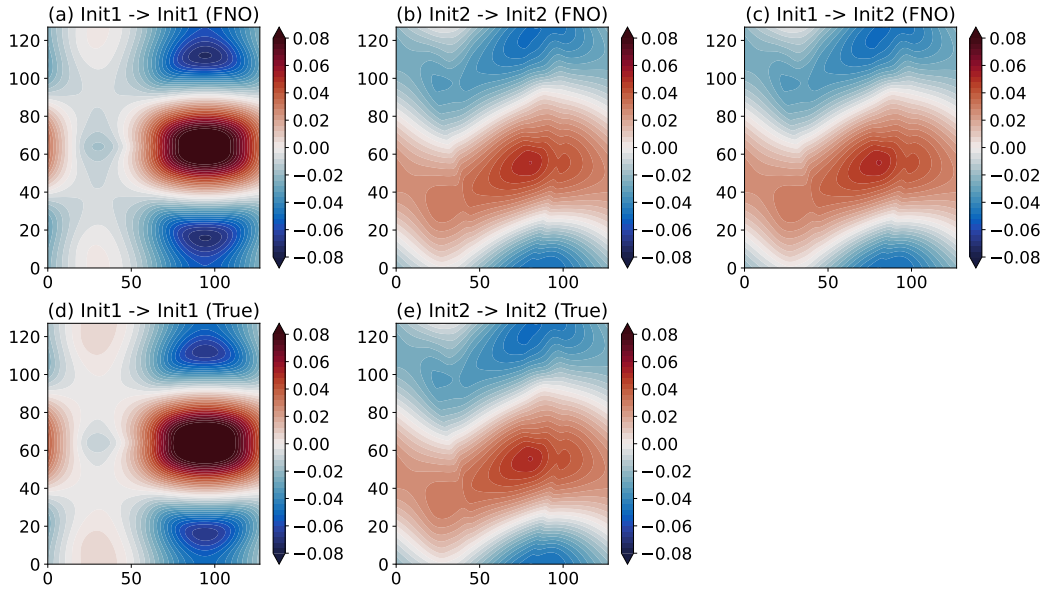

Figure 11. Zero-shot learning for different initial conditions of x-velocity. (a) and (b) are the predictions of x-vel where the training data and the testing data are from the same simulations. (d) and (e) are the corresponding ground truth. (c) is the generalizing prediction where the training data and testing data are from the two simulations with different initial conditions. The label "init1/2" before the arrow in the subtitle denotes the training data, while the label after the arrow represents the testing, or prediction, data. The normalized L2 error between the FNO prediction (c) and the ground truth (e) is 0.0001.

Among all models, the FNO model is the only one capable of performing zero-shot *super-resolution*, which means a model trained on low resolutions can also make prediction on high resolution grids. In comparison, UNet and ResNet models cannot adapt to resolutions which they are not trained on. In this study, we utilized the FNO model trained on a R64 resolution and applied it to R128 and R256 resolutions. The results, as shown in Figures 13 and 14, revealed an excellent match with the ground truth. Regarding x-

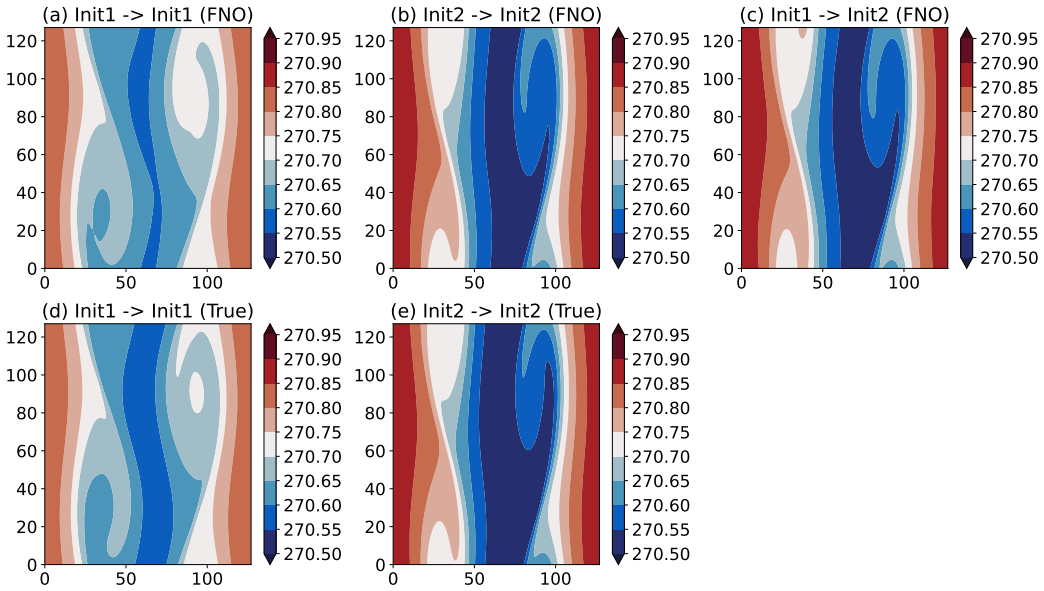

Figure 12. Same as Figure 11, but for temperature. The normalized L2 error between the FNO prediction (c) and the ground truth (e) is 0.094.

velocity, the normalized L2 errors achieve 0.0004 and 0.0007 for R128 and R256 super-resolution, respectively. As for temperature, they achieve 0.038 and 0.036 for R128 and R256 super-resolution, respectively. These results demonstrate that the accuracy of super-resolution is comparable to that of training from the higher resolution (Table 1).

These findings are significant as they can facilitate a more comprehensive understanding of turbulent clouds, especially for the large-scale turbulent eddies, by extending the PR-DNS domain to larger domains. PR-DNS models with larger domains can better represent larger turbulent eddies and their influences on cloud microphysics, especially for turbulent clouds with large Reynold numbers. Such multiscale interactions could be critical for determining such macro-cloud properties such as lifetime and physical sizes.

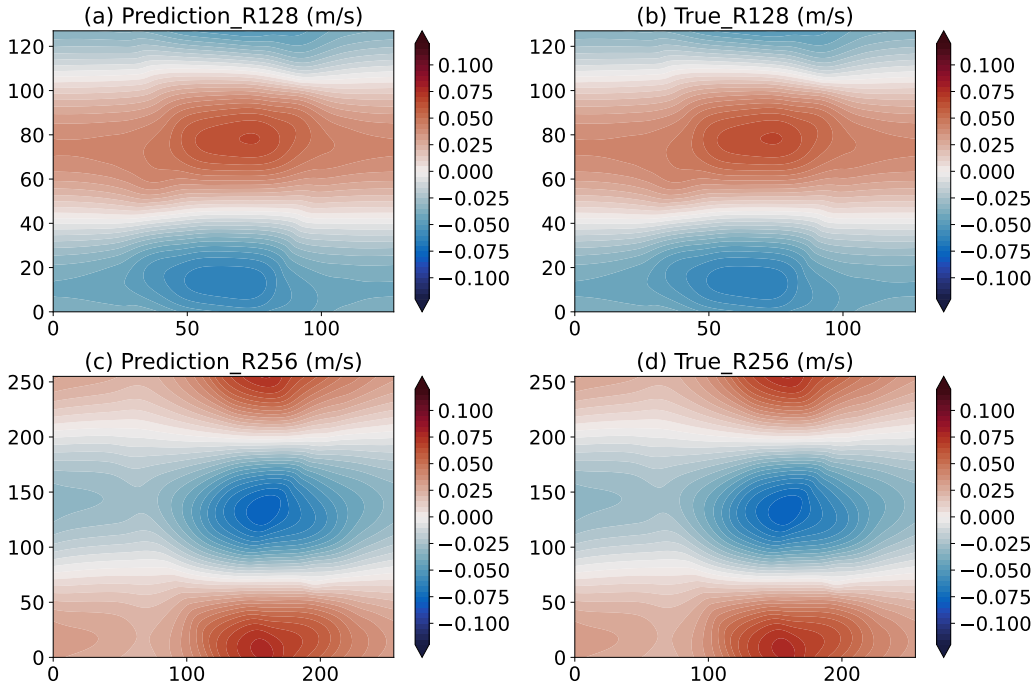

Figure 13. Super-resolution to R128 (a, b) and R256 (c, d) using the FNO model trained on the R64 resolution in terms of x-velocity.

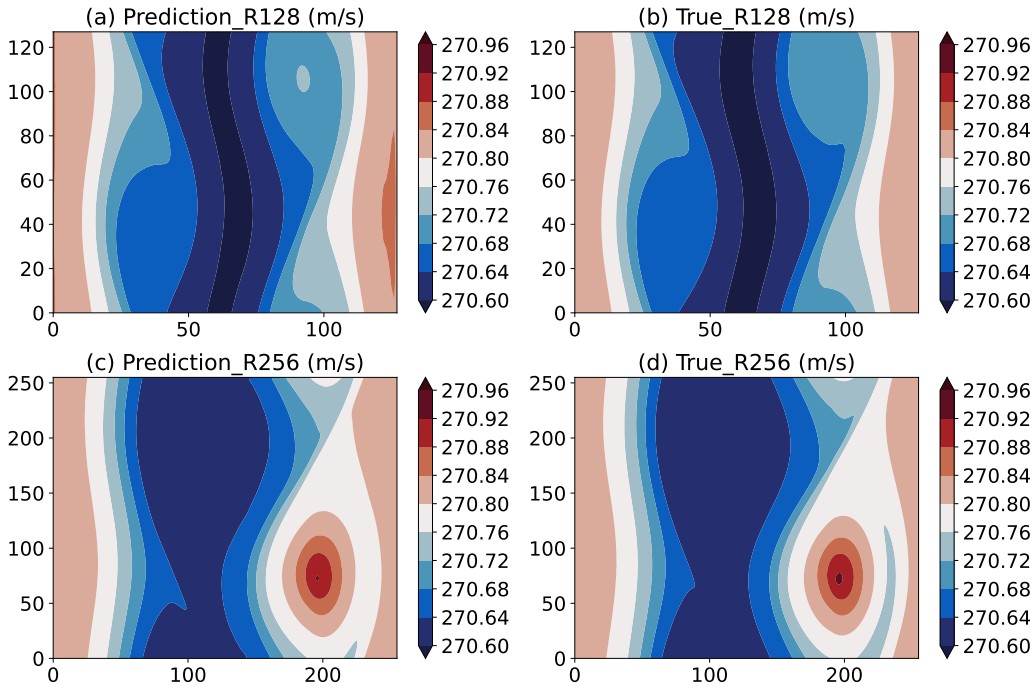

Figure 14. Super-resolution to R128 (a, b) and R256 (c, d) using the FNO model trained on the R64 resolution in terms of temperature.

5 Discussion and conclusion

In this study, we first present a digital twin model of the complex PR-DNS developed by use of the Fourier neural operator (FNO) method. PR-DNS is crucial for enhancing our understanding of the intricate aerosol-cloud-turbulence interactions at the process level. However, it incurs a substantial high computational cost due to the long simulations for large domains with ultra-high resolutions. The digital twin of the numerical PR-DNS model serves as a surrogate for the original physics-based model, and it typically relies on the output generated by the original PR-DNS simulations. The utilization of digital twins is essential because it enables the extreme-high-resolution simulation in a controlled and cost-effective manner. As a consequence, the digital twins can facilitate the understanding of complex physical systems under different scenarios, such as varying initial conditions and resolutions.

While the FNO model is state-of-the-art, there is still room for exploration in future work. First, this study is focused on the 2D spatial domain, while in reality, the original PR-DNS solver can run 3D simulations. Hence, we aim to develop a 3D FNO model. Second, it's worth noting that this study only focuses on the Eulerian part, specifically the velocity and temperature. It will be important to expand this study to the moisture variables, such as supersaturation and water vapor mixing ratio. Additionally, besides the Eulerian part, the Lagrangian part, such as particle motion and growth, should also be considered.

In our work, an 'offline' learning method is employed, where the testing dataset is generated by the original PR-DNS. However, for more realistic usage, an 'online' learning method should be used, where the current input of the deep learning model is derived from the previous output of the deep learning operator. However, it should be noted that in the online approach, the prediction errors may accumulate and significantly affect the accuracy of the simulation. To address this issue, the error can be constrained by the original PR-DNS model or through the use of the Physics-Informed Neural Network (PINN) method (Raissi et al., 2019), which incorporates the governing equations as constraints.

Finally, in this study, the time step of FNO is the same as the original PR-DNS. Due to the use of different solvers, we will explore appropriate steps for the new deep learning solver in future work.

6 Open Research

Data Availability Statement

The deep learning models, including FNO, ResNet and UNet can be archived at https://doi.org/10.5281/zenodo.8077871. The PR-DNS time-step simulations at various initial conditions and various resolutions for training the deep learning models can be available at https://doi.org/10.5281/zenodo.8077871. The PR-DNS models can be accessed at https://github.com/PR-DNS/PR_DNS_base.

Acknowledgments
This work is supported by the Brookhaven National Laboratory's Laboratory Directed Research and Development (LDRD) Project 22065.

References

Andrejczuk, M., Grabowski, W. W., Malinowski, S. P., & Smolarkiewicz, P. K. (2004). Numerical simulation of cloud–clear air interfacial mixing. *Journal of the atmospheric sciences*, *61*(14), 1726–1739.

Andrejczuk, M., Grabowski, W. W., Malinowski, S. P., & Smolarkiewicz, P. K. (2006). Numerical simulation of cloud–clear air interfacial mixing: Effects on cloud microphysics. *Journal of the atmospheric sciences*, *63*(12), 3204–3225.

Andrejczuk, M., Grabowski, W. W., Malinowski, S. P., & Smolarkiewicz, P. K. (2009). Numerical simulation of cloud–clear air interfacial mixing: Homogeneous versus inhomogeneous mixing. *Journal of the Atmospheric Sciences*, *66*(8), 2493–2500.

Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., ... others (2019). Petsc users manual.

Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., & Kaushik, S. (2019). Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, *64*, 525–545.

Chen, S., Bartello, P., Yau, M., Vaillancourt, P., & Zwijsen, K. (2016). Cloud droplet collisions in turbulent environment: Collision statistics and parameterization. *Journal of the Atmospheric Sciences*, *73*(2), 621–636.

Chen, S., Xue, L., & Yau, M.-K. (2020). Impact of aerosols and turbulence on cloud droplet growth: an in-cloud seeding case study using a parcel–dns (direct numerical simulation) approach. *Atmospheric Chemistry and Physics*, *20*(17), 10111–10124.

Chen, S., Yau, M., & Bartello, P. (2018). Turbulence effects of collision efficiency and broadening of droplet size distribution in cumulus clouds. *Journal of the Atmospheric Sciences*, *75*(1), 203–217.

Chen, S., Yau, M.-K., Bartello, P., & Xue, L. (2018). Bridging the condensation–collision size gap: a direct numerical simulation of continuous droplet growth in turbulent clouds. *Atmospheric Chemistry and Physics*, *18*(10), 7251–7262.

Falgout, R. D., & Yang, U. M. (2002). hypre: A library of high performance preconditioners. In *Computational science—iccs 2002: International conference amsterdam, the netherlands, april 21–24, 2002 proceedings, part iii* (pp. 632–641).

Fan, J., Leung, L. R., Rosenfeld, D., Chen, Q., Li, Z., Zhang, J., & Yan, H. (2013). Microphysical effects determine macrophysical response for aerosol impacts on deep convective clouds. *Proceedings of the National Academy of Sciences*, *110*(48), E4581–E4590.

Gao, Z., Liu, Y., Li, X., & Lu, C. (2018). Investigation of turbulent entrainment-mixing processes with a new particle-resolved direct numerical simulation model. *Journal of Geophysical Research: Atmospheres*, *123*(4), 2194–2214.

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).

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., & Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, *118*(21), e2101784118.

Kumar, B., Bera, S., Prabha, T. V., & Grabowski, W. W. (2017). Cloud-edge mixing: Direct numerical simulation and observations in i ndian m onsoon clouds. *Journal of Advances in Modeling Earth Systems*, *9*(1), 332–353.

Kumar, B., Janetzko, F., Schumacher, J., & Shaw, R. A. (2012). Extreme responses of a coupled scalar–particle system during turbulent mixing. *New Journal of Physics*, *14*(11), 115020.

Kumar, B., Schumacher, J., & Shaw, R. A. (2013). Cloud microphysical effects of turbulent mixing and entrainment. *Theoretical and Computational Fluid Dynamics*, *27*, 361–376.

Kumar, B., Schumacher, J., & Shaw, R. A. (2014). Lagrangian mixing dynamics at the cloudy–clear air interface. *Journal of the Atmospheric Sciences*, *71*(7), 2564–2580.

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

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020b). Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*.

Liu, Y. (2019). Introduction to the special section on fast physics in climate models: Parameterization, evaluation, and observation. *Journal of Geophysical Research: Atmospheres*, *124*(15), 8631–8644.

Liu, Y., Yau, M.-K., Shima, S.-i., Lu, C., & Chen, S. (2023). Parameterization and explicit modeling of cloud microphysics: Approaches, challenges, and future directions. *Advances in Atmospheric Sciences*, *40*(5), 747–790.

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.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

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.

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).

Sullivan, P. P., McWilliams, J. C., & Moeng, C.-H. (1994). A subgrid-scale model for large-eddy simulation of planetary boundary-layer flows. *Boundary-Layer Meteorology*, *71*, 247–276.

Vaillancourt, P., Yau, M., Bartello, P., & Grabowski, W. W. (2002). Microscopic approach to cloud droplet growth by condensation. part ii: Turbulence, clustering, and condensational growth. *Journal of the atmospheric sciences*, *59*(24), 3421–3435.

Vaillancourt, P. A., & Yau, M. (2000). Review of particle-turbulence interactions and consequences for cloud physics. *Bulletin of the American Meteorological Society*, *81*(2), 285–298.

Van Heerwaarden, C. C., Van Stratum, B. J., Heus, T., Gibbs, J. A., Fedorovich, E., & Mellado, J. P. (2017). Microhh 1.0: A computational fluid dynamics

code for direct numerical simulation and large-eddy simulation of atmospheric boundary layer flows. *Geoscientific Model Development*, *10*(8), 3145–3165.

Xie, C., Li, K., Ma, C., & Wang, J. (2019). Modeling subgrid-scale force and divergence of heat flux of compressible isotropic turbulence by artificial neural network. *Physical Review Fluids*, *4*(10), 104605.