Prometheus: Out-of-distribution Fluid Dynamics Modeling with Disentangled Graph ODE

Hao Wu [1 2] **Huiyuan Wang** [3 4] **Kun Wang** [2] **Weiyan Wang** [1] **Changan Ye** [1] **Yangyu Tao** [1]
Chong Chen [5] **Xian-Sheng Hua** [5] **Xiao Luo** [6]

Abstract

Fluid dynamics modeling has received extensive attention in the machine learning community. Although numerous graph neural network (GNN) approaches have been proposed for this problem, the problem of out-of-distribution (OOD) generalization remains underexplored. In this work, we propose a new large-scale dataset *Prometheus* which simulates tunnel and pool fires across various environmental conditions and builds an extensive benchmark of 13 baselines, which demonstrates that the OOD generalization performance is far from satisfactory. To tackle this, this paper introduces a new approach named Disentangled Graph ODE (DGODE), which learns disentangled representations for continuous interacting dynamics modeling. In particular, we utilize a temporal GNN and a frequency network to extract semantics from historical trajectories into node representations and environment representations respectively. To mitigate the potential distribution shift, we minimize the mutual information between invariant node representations and the discretized environment features using adversarial learning. Then, they are fed into an environment-aware graph ODE framework, which models the evolution using neighboring nodes and dynamical environmental context. In addition, we enhance the stability of the framework by perturbing the environment features to enhance robustness. Experiments validate the effectiveness of DGODE compared with state-of-the-art approaches.

[1]Machine Learning Platform Department, Tencent [2]University of Science and Technology of China [3]The Center for Health AI and Synthesis of Evidence (CHASE), University of Pennsylvania [4] Department of Biostatistics, Epidemiology, and Informatics, University of Pennsylvania Perelman School of Medicine [5]Terminus Group [6]University of California, Los Angeles. Correspondence to: Xiao Luo <xiaoluo@cs.ucla.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Computational fluid dynamics modeling (Kochkov et al., 2021; Obiols-Sales et al., 2020; Steeven et al., 2024; Li et al., 2021; Wu et al., 2023c; Wu et al.) is an important problem in fluid mechanics, which can facilitate our understanding of fluid flows (Li et al., 2023a; Yu et al., 2024). Recently, a range of machine learning approaches have been developed to solve this problem (Pfaff et al., 2021; Li et al., 2023b; Shao et al., 2022; Sanchez-Gonzalez et al., 2020; Han et al., 2022a; Deng & Hooi, 2021). They usually construct geometric graphs to model the relationships between neighborhood observation points and then adopt graph neural networks (GNNs) to model fluid dynamics, which follow the message passing mechanism (Kipf & Welling, 2017; Veličković et al., 2018) by aggregating neighborhood information of each node for interactive updating.

Although these approaches have achieved some progress in fluid dynamics modeling (Yu et al., 2023; Huang et al., 2023a; Luo et al., 2024), they usually assume that training and test data come from the same distribution (Pfaff et al., 2021; Shao et al., 2022; Sanchez-Gonzalez et al., 2020; Lippe et al., 2023), which is not practical in the real-world analysis since fluid mechanisms always face different environments resulting from system parameters such as temperature, viscosity and pressure (Baradel et al., 2019; Sanchez-Gonzalez et al., 2020; Huang et al., 2023b). Existing machine learning algorithms would suffer from inferior out-of-distribution (OOD) performance (Hendrycks et al., 2021). Therefore, in this work, we study the problem of OOD fluid dynamics modeling, which aims to learn data-driven models generalized well on systems with different parameters. However, there are limited large-scale fluid dynamics datasets and benchmarks for this practical and emerging problem, which could hinder from algorithm research and development.

In this work, we first build a large-scale OOD fluid dynamics dataset *Prometheus* using extensive fire simulations. In particular, both tunnel and pool fires are modeled using fire dynamics simulators, which solve Navier-Stokes equations in fire scenarios. More importantly, our simulators consider 25 different environments with varying parame-

ters including HRR and ventilation speeds, which results in well-designed datasets for OOD generalization evaluation. Totally, we generate 4.8TB of raw data compressed into 340GB and construct the benchmark with 12 popular dynamics modeling baselines.

As a second contribution, we propose an effective solution to this problem. Our motivation is to answer the following two research questions: (1) *How to handle distribution shifts from different system parameters for fluid dynamics modeling?* Different system parameters could result in distribution shifts across dynamical systems and thus spurious correlations involved in trajectory predictions. Existing OOD generalization problems (Liu et al., 2021a; Gui et al., 2023) usually capture invariant patterns across several environments using environment labels, which can help to remove spurious correlations. However, we would encounter a large variety of different environments, and their labels are usually unavailable for fluid dynamics due to their high complexity. (2) *How to model continuous complicated interacting dynamics across different environments?* Recent GNN approaches (Pfaff et al., 2021; Shao et al., 2022; Sanchez-Gonzalez et al., 2020) usually leverage the current states for next-time predictions without consideration of the environmental context, which could continuously influence the evolution mechanisms. Moreover, since fluid dynamics is intrinsically continuous, it is highly expected to incorporate environmental context into a unified and continuous trajectory forecasting framework.

In this paper, we propose a novel framework termed Disentangled Graph ODE (DGODE) for OOD fluid dynamics modeling. The core of our DGODE is to learn disentangled representations for capturing continuous dynamics. Concretely, we utilize a temporal GNN and a frequency network, which learns spatio-temporal relationships and frequency semantics from complementary views for node features and environment features. To mitigate the OOD distribution shift across different environments, we aim to minimize the mutual information between invariant node features and environment features. To achieve this, we stratify all environmental features into discrete pieces using **vector quantization** and minimize the upper bound of mutual information by **adversarial learning**. To model the continuous complicated interacting dynamics under different environments, we incorporate all the node features and environment features into a graph ODE framework, which models the evolution of latent states of interacting nodes and the environment simultaneously. Here, node states are driven by interacting neighboring nodes and the environment while environment states always evolve to simulate dynamical influence. Finally, a decoder is adopted to output the future trajectories and we also perturb the environment features to enhance the stability of node states *w.r.t* different environments for generalization robustness. Extensive experiments on various benchmarks showcase the superiority of the DGODE compared to existing SOTAs.

The contribution of this paper can be summarized as follows: **(I)** We propose a new large-scale dataset and an extensive benchmark of 12 baselines for OOD fluid dynamics modeling. **(II)** We develop a novel approach named DGODE for this problem. Our DGODE learns disentangled node and environment representations to mitigate the risk of distribution shift and incorporates them into a continuous graph ODE framework to model interacting dynamics across different environments. **(III)** Extensive experiments validate the effectiveness of the proposed DGODE in comparison to a wide range of approaches. Our Prometheus dataset can be found at the following link: https://huggingface.co/datasets/easylearning/Prometheus.

2. Benchmark

Problem Definition. Given a fluid dynamical system, we characterize the observation states and interaction at the t-th step using a graph $G^t = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}^t)$. Each node represents an observation point, $\mathcal{E}$ collects all the edges, and $\boldsymbol{X}^t$ denotes the attribute matrix. In our settings, we employ the historical trajectories $X^H = X^{1:T_{obs}}$ and aim to learn a model to predict the future trajectories $X^F = X^{T_{obs}+1:T}$. In our problem, the fluid dynamics can be governed by a range of different hidden system parameters, which results in different environments. These environments would lead to potential distribution shifts in historical trajectories to degrade the forecasting performance during inference.

Purpose. We built ***Prometheus*** to meet the growing demand for deep learning technologies and fluid dynamics data. This dataset integrates advanced methodologies from the field of engineering, with a special focus on precise and efficient data analysis and inference on irregular grid structures. As shown in Figure 1, we use Computational Fluid Dynamics (CFD) technology to focus on simulating *tunnel fires* and *pool fires*. By finely adjusting a variety of physical parameters, we can simulate different environmental conditions. This helps us recreate complex and challenging turbulence phenomena. Moreover, this approach is very beneficial for conducting Out-Of-Distribution (OOD) tests.

2.1. Proxy Task Configuration

In this section, we provide a detailed introduction to the design of proxy tasks. This includes the geometric shape of the work condition, the design of physical parameters, and the grid situation in the computational domain, among other aspects. The specific descriptions are shown below:

Tunnel Fires. In this study, we design a proxy task to investigate tunnel fire dynamics, as shown in Figure 1(a), building a $6 \times 6 \times 100$ m simulation tunnel with a consis-

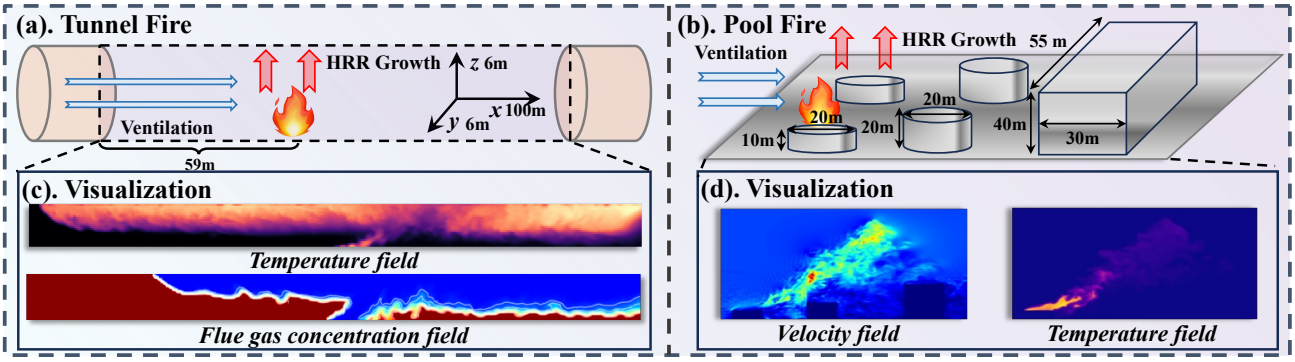


Figure 1. **(a)**. Tunnel Design: A 100 m long, 6 m wide and high tunnel, with a focus on 2D flow data at y=3 m, using sensors for temperature and visibility. **(b)**. Industrial Park Tanks: Three tanks, each 20 m in diameter, with heights of 10 m, 20 m, and a building of 55 m length, 40 m width, and 30 m height. **(c)**. Tunnel Visualization: Displaying temperature and visibility distribution. **(d)**. Fire Dynamics: Visualizing velocity and temperature in a pool fire to study its dynamics.

tent landscape and a fixed fire source 59 meters from the entrance. We vary the heat release rate (HRR) and ventilation speed to simulate different fire scenarios, observing smoke stratification, flame spread, and diverse burning patterns. We equip the tunnel with 32×480 sensors to measure temperature and flue gas concentration, creating 25 environmental combinations with varying HRR and ventilation speeds. Additionally, we introduce vent parameters in five settings as disturbances, forming a 30-condition dataset for tunnel fire combustion analysis. These 30 environments are denoted as $a_1, a_2, a_3, ..., a_{30}$.

Pool Fires. Next, as shown in Figure 1(b), we build a fire model for a tank storage area, our goal is to examine its impact on the surroundings. The model spans a 150×100 m area and includes tanks, buildings, and pool fires. We position the simulated fire source above the leftmost tank, 5 meters from other structures. Our study finely adjusts the HRR and ventilation speed to assess different variables. We focus on analyzing changes in flame contours, temperature, and internal velocity under various HRR conditions, and how ventilation affects flame length and tilt angle. The study also evaluates the thermal impact of a fire on nearby buildings and tanks. We deploy a sensor grid in and around the model area to measure temperature and speed, setting 25 environmental combinations of HRR and ventilation speeds for simulating different fire scenarios. Sensor data helps us understand the temperature distribution and velocity fields in the area during tank fires. We denote these 25 environments as $b_1, b_2, b_3, ..., b_{25}$.

2.2. Numerical Simulation

In combustion dynamics modeling, we use FDS (Fire Dynamics Simulator) based on the Navier-Stokes equations to simulate thermal combustion phenomena accurately. These equations form the foundation of fluid dynamics, describing gas and liquid flow dynamics. To simulate fire scenarios better, we adjust and expand the Navier-Stokes equations to consider heat transfer, combustion processes, and smoke propagation. This method generates 4.8TB of raw data with a grid precision of 2 million. We then downsample the data and compress it to 340GB. More details and data processing information are available in the Appendix E.

2.3. Physical Field Visualization

In this section, we detail the simulation's 10 physical variables in tunnel and building pool fire scenarios, as illustrated in Figure 6, divided into two categories for in-depth explanation. ***(1). Pool Fires***: We analyze pool fires from the front and top views. The front view shows Z-axis velocity (v_z), X-axis velocity (v_{x1}), and their resultant speed ($v_{speed1} = \sqrt{{v_z}^2 + v_{x1}^2}$), along with temperature (tem_{pool1}). The top view presents Y-axis velocity (v_y), X-axis velocity (v_{x2}), their resultant speed ($v_{speed2} = \sqrt{{v_y}^2 + v_{x2}^2}$), and temperature ($tem_{pool2}$). ***(2). Tunnel Fires***: For tunnel fires, we examine the tunnel's longitudinal section, focusing on temperature (tem_{tunnel}) and flue gas concentration ($flue_{tunnel}$), to understand their distribution inside the tunnel.

3. Methodology

3.1. Framework Overview

This work studies the problem of OOD fluid dynamics modeling, which is challenging due to distribution shifts across different environments and complicated interacting dynamics. Here, we propose a new approach named DGODE for this problem, which can learn disentangled representations for continuous dynamics modeling. In particular, we involve a temporal GNN and a frequency network to extract node features and environment features. To tackle the OOD shift, we stratify environments using vector quantization and conduct representation disentanglement by mutual information

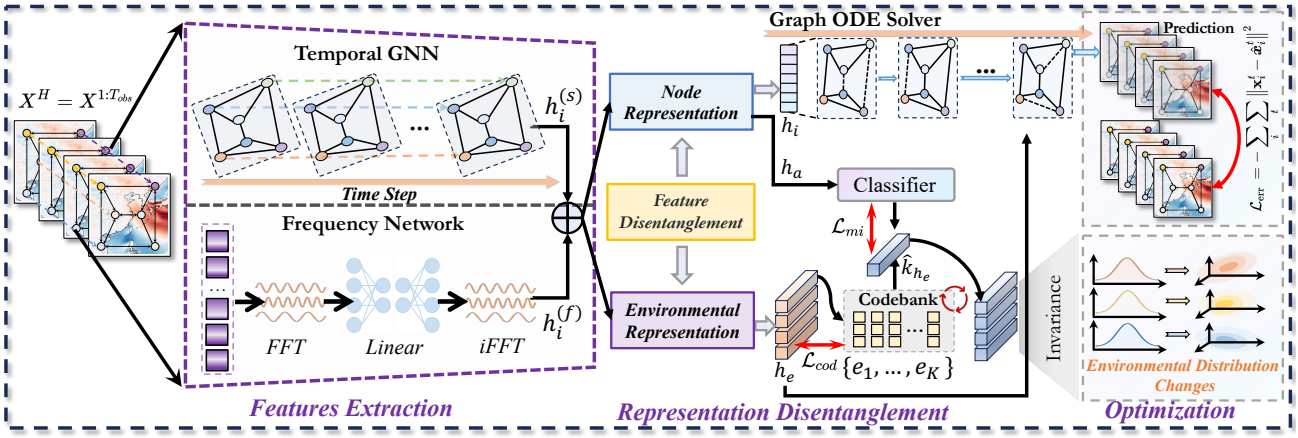


Figure 2. An Overview of DGODE. Our DGODE utilizes a temporal GNN and a frequency network to generate node features and environment features. We adopt vector quantization to discretize environment features, which would be disentangled with node features by mutual information minimization. These features would be fed into an environment-aware graph ODE to model the evolution of dynamical systems. We also minimize the variance of node features across different environments for enhanced generalization capacity.

minimization. To model the complicated interacting dynamics under different environments, we incorporate all the node features and environment features into a graph ODE framework, which models the evolution using states of interacting nodes and the environment simultaneously. We also introduce a regularization term to enhance the stability of our framework to different environments. An overview of our DGODE can be found in Figure 2.

3.2. Learning Disentangled Representations from Historical Trajectories

The major obstacle is that different environments would bring in distribution shift (Rämä & Sipilä, 2017; Luo et al., 2023a), which would generate spurious correlations between historical trajectories and future predictions. To tackle this, we introduce representation disentanglement, which decomposes historical trajectories into invariant node representations and environment representations with mutual information minimization. Here, a temporal GNN (Huang et al., 2021) and a frequency network are adopted to generate node features and environment features from complementary views. More importantly, we stratify all the environment features using vector quantization representation and then ensure representation disentanglement by mutual information minimization (Belghazi et al., 2018), which can mitigate the influence of environments on node representations and help to capture invariant patterns in OOD scenarios.

Feature Extraction. In detail, we first utilize a temporal GNN to summarize the node representations from historical trajectories, which updates temporal node representations using their related nodes. In formulation, given a temporal node representation $\boldsymbol{h}_i^{t,(l)}$ at the l-th layer, the update rule can be formulated as:

$$\boldsymbol{h}_i^{t,(l)} = \phi^e([\boldsymbol{h}_i^{t,(l-1)}, \boldsymbol{h}_i^{t-1,(l-1)}, \boldsymbol{N}_i^{t,(l-1)}]), \quad (1)$$

where $\boldsymbol{N}_i^{t,(l-1)} = \mathrm{AGG}(\boldsymbol{h}_j^{t,(l-1)} | j \in \mathcal{N}(i))$ is the neighborhood representation and $\mathrm{AGG}(\cdot)$ is the aggregation operator. After stacking L layers, we summarize all the temporal node representations into spatial node features for downstream forecasting by:

$$\boldsymbol{h}_i^{(s)} = \mathrm{SUM}(\{\boldsymbol{h}_i^{t,(L)} | t = 1, 2, ..., T_{obs}\}), \quad (2)$$

where $\mathrm{SUM}(\cdot)$ denotes the summarization operator. Besides temporal GNNs, we leverage the frequency domain to enhance the representation learning, which can explore the periodic information as a complementary. To be specific, we utilize a Fast Fourier Transform (FFT) (Li et al., 2021) to transform historical data into the frequency domain, and then utilize a feed-forward network (FFN) to feature updates, followed by an inverse Fast Fourier Transform (iFFT). Finally, these reconstructed features would be reshaped frequency-based node features $\boldsymbol{h}_i^{(f)}$. In formulation, we have:

$$\{\boldsymbol{h}_i^{(f)}\}_{i \in V} = \mathrm{Re}(\mathrm{iFFT}(\mathrm{FFN}(\mathrm{FFT}(G^{1:T_{obs}}))), \quad (3)$$

where $\mathrm{Re}(\cdot)$ is to convert the reconstructed feature maps into a set of node features. Finally, we concatenate spatial node features and frequency-based node features generated from complementary views into final node features as:

$$\boldsymbol{h}_i = [\boldsymbol{h}_i^{(s)}, \boldsymbol{h}_i^{(f)}]. \quad (4)$$

To generate environmental representations, we utilize the same backbone with different parameters for node features

$\boldsymbol{h}_i'$, followed by a summarization operator for environment features. In formulation, we have:

$$\boldsymbol{h}_e = \text{SUM}(\{\boldsymbol{h}_i' | i \in V\}). \tag{5}$$

Representation Disentanglement. To enhance the invariance of node features across different environments, we disentangle $\{\boldsymbol{h}_i^{(f)}\}_{i\in V}$ and $\boldsymbol{e}$ using mutual information minimization. However, $I(\{\boldsymbol{h}_i\}_{i\in\mathcal{V}}; \boldsymbol{h}_e)$ is difficult to obtain. To tackle this, we stratify all the environments into discrete pieces by introducing the vector quantization strategy. Here, we have a codebook $\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_K\}$ and map each environment feature $\boldsymbol{h}_e$ into its closed one by:

$$\text{VQ}(\boldsymbol{h}_e) = \boldsymbol{e}_{\hat{k}}, \text{ where } \hat{k}_{\boldsymbol{h}_e} = \text{argmin}_k \left\| \boldsymbol{h}(\boldsymbol{e}) - \boldsymbol{e}_k \right\|_2, \tag{6}$$

The quantization (Zhou et al., 2023; Xia et al., 2024; Van Den Oord et al., 2017) can decrease the complexity of our model, which can facilitate our mutual information maximization (Huang et al., 2023b) as well as the stability maximization in Eqn. 20. After stratification, given a graph sample, we concatenate all the node features into $\boldsymbol{h}_a$ and introduce a classifier g_ψ to predict the environment $p(\hat{k}_{\boldsymbol{h}_e}|\boldsymbol{h}_a)$. Then, we can estimate the upper bound of mutual information, which can be written as:

$$\mathcal{L}_{mi} = \mathbb{E}_{p(\boldsymbol{h}_a, \hat{k}_{\boldsymbol{h}_e})}[p(\hat{k}_{\boldsymbol{h}_e}|\boldsymbol{h}_a)] - \mathbb{E}_{p(\boldsymbol{h}_a)}\mathbb{E}_{p(\hat{k}')}[p(\hat{k}'|\boldsymbol{h}_a)]], \tag{7}$$

To model $p(\hat{k}_{\boldsymbol{h}_e}|\boldsymbol{h}_a)$, we maximize Eqn. 7 with respect to classifier parameters ψ for accurate prediction. Then, we minimize Eqn. 7 with respect to network parameters θ for disentanglement. The alternative optimization is conducted in a standard adversarial learning paradigm (Luo et al., 2023c). The minimal-extremal formula is:

$$\begin{aligned}\min_\theta \max_\psi \mathcal{L}_{mi} = \min_\theta \max_\psi \Big(&\mathbb{E}_{p(h_a, \hat{k}_{h_e})}[p(\hat{k}_{h_e}|h_a)] \\ &-\mathbb{E}_{p(h_a)}\mathbb{E}_{p(\hat{k}')}[p(\hat{k}'|h_a)]\Big)\end{aligned} \tag{8}$$

Eqn. 8 aims to achieve representation disentanglement through adversarial learning by adjusting classifier parameters ψ to maximize environmental predictive accuracy and network parameters θ to minimize mutual information, thereby promoting feature disentanglement.

3.3. Environment-aware Graph ODE

After generating disentangled representations for both node representations and environment representations, we aim to develop an effective framework for dynamical system modeling, which is challenging due to complicated continuous interacting dynamics (Chen et al., 2024a; Luo et al., 2024; Huang et al., 2023b; 2022) and environmental influence. To tackle this, we introduce an environment-aware graph ODE framework, which models the evolution of node states and environment states simultaneously.

In particular, the initial node states and environments for our ODE model are from their features, i.e., $\boldsymbol{z}_i^0 = \boldsymbol{h}_i$ and $\boldsymbol{z}_e^0 = \boldsymbol{h}_e$. Then, the interacting dynamics can be modeled by a graph ODE as:

$$\frac{d\boldsymbol{z}_i^t}{dt} = \sigma\left(\sum_{j\in\mathcal{N}^s(i)} \boldsymbol{w}_{ij}\boldsymbol{z}_j^s\boldsymbol{W}_1 + \boldsymbol{z}_e\boldsymbol{W}_2\right), \tag{9}$$

$$\frac{d\boldsymbol{z}_e^t}{dt} = \sigma\left(\sum_{i\in\mathcal{V}} \boldsymbol{z}_j^s\boldsymbol{W}_3 + \boldsymbol{z}_e\boldsymbol{W}_4\right), \tag{10}$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_3$ and $\boldsymbol{W}_4$ are learnable weight matrix. $\boldsymbol{w}_{ij} = \frac{\hat{\boldsymbol{A}}_{ij}}{\sqrt{\hat{\boldsymbol{D}}_i\hat{\boldsymbol{D}}_j}}$ is the fixed weight from adjacency matrix where $\hat{\boldsymbol{D}}$ and $\hat{\boldsymbol{A}}$ denote the degree matrix and the adjacency matrix of the graph with self-loop. σ is an activation function. Eqn. 9 models the evolution of hidden states using neighboring nodes and the environment state. Moreover, Eqn. 10 models the evolution of the environment state using all the nodes in the system, which can characterize the dynamical influence of environments. In comparison to previous approaches (Huang et al., 2020; Chen et al., 2018), we introduce environment states into the graph ODE framework to model complicated OOD fluid dynamics. Our environment-aware ODE framework can be viewed as adding a virtual node connected with all nodes in the graph. Finally, we can utilize a decoder $\phi^d(\cdot)$ to produce the future trajectories as:

$$\hat{\boldsymbol{x}}_i^t = \phi^d(\boldsymbol{z}_i^t). \tag{11}$$

Theoretical Analysis. In this part, we will theoretically show the necessity of incorporating the environment states into the system. Consider the case where we have the p-dimension environment vector $\boldsymbol{z}_e(t) = (z_{e1}(t), \ldots, z_{ep}(t))^T$ and p node states $\boldsymbol{z}(t) = (z_1(t), \ldots, z_p(t))^T$. Suppose that the interacting dynamics of node states and environments follows a system of linear ordinary differential equations,

$$\begin{aligned}\frac{d\boldsymbol{z}(t)}{dt} &= \boldsymbol{W}_1^*\boldsymbol{z}(t) + \boldsymbol{W}_2^*\boldsymbol{z}_e(t), \quad \boldsymbol{z}(0) = \boldsymbol{h}, \\ \frac{d\boldsymbol{z}_e(t)}{dt} &= \boldsymbol{W}_3^*\boldsymbol{z}(t) + \boldsymbol{W}_4^*\boldsymbol{z}_e(t), \quad \boldsymbol{z}_e(0) = \boldsymbol{h}_e,\end{aligned} \tag{12}$$

where $\boldsymbol{W}_1^*$, $\boldsymbol{W}_2^*$, $\boldsymbol{W}_3^*$, and $\boldsymbol{W}_4^*$ are target parameters. Define $\boldsymbol{Z}(t) = (\boldsymbol{z}^T(t), \boldsymbol{z}_e^T(t))^T$. Thus, the ODE for the concatenated vector can be written compactly below,

$$\frac{d\boldsymbol{Z}(t)}{dt} = \boldsymbol{W}^*\boldsymbol{Z}(t), \tag{13}$$

where $\boldsymbol{W}^* = \begin{pmatrix}\boldsymbol{W}_1^*, \boldsymbol{W}_2^* \\ \boldsymbol{W}_3^*, \boldsymbol{W}_4^*\end{pmatrix}$. The dynamics of the nodes and environment states follow:

$$\boldsymbol{Z}(t) = e^{t\boldsymbol{W}^*}\boldsymbol{Z}(0), \quad \boldsymbol{Z}(0) = (\boldsymbol{h}^T, \boldsymbol{h}_e^T)^T. \tag{14}$$

We assume that both $\boldsymbol{z}(t)$ and $\boldsymbol{z}_e(t)$ are measured with independent Gaussian errors at finite time points; that is,

$$\tilde{\boldsymbol{Z}}(t_j) = \boldsymbol{Z}(t_j) + \varepsilon_j, \tag{15}$$

where $\varepsilon_j \sim N(0, \sigma^2 \boldsymbol{I})$, and without loss of generality $t_j = j/n, j = 1, \ldots, n$.

Theorem 3.1. *Under model Eqn. 14–Eqn. 15 where $\boldsymbol{W}^*$ and $\boldsymbol{Z}(0)$ both have strictly positive components, if we ignore the environment states $\boldsymbol{z}_e$; that is, we specify a wrong working model:*

$$\frac{d\boldsymbol{z}(t)}{dt} = \boldsymbol{W} z(t), \quad \boldsymbol{z}(0) = \boldsymbol{h}, \tag{16}$$

then the resulting least square estimator,

$$\widehat{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}} \frac{1}{n} \sum_{j=1}^{n} \|\tilde{\boldsymbol{z}}(t_j) - \boldsymbol{z}(t_j; \boldsymbol{W})\|_2^2, \tag{17}$$

is inconsistent to $\boldsymbol{W}_1^*$ even when $n \to \infty$, where $\boldsymbol{z}(t; \boldsymbol{W}) = e^{t\boldsymbol{W}} \boldsymbol{h}$ denotes the solution to the ODE Eqn. 16.

The proof of Theorem 3.1 can be found in Appendix A.

3.4. Optimization

To optimize the whole framework, we minimize the training error compared with the ground truth as follows:

$$\mathcal{L}_{err} = -\sum_i \sum_t \left\| \mathbf{x}_i^t - \hat{\boldsymbol{x}}_i^t \right\|^2. \tag{18}$$

To optimize the codebook with standard gradient descent, we integrate the stopgradient operator $sg(\cdot)$ (Van Den Oord et al., 2017), which is an identity operator during forwarding the network and cuts off the gradient computation during back propagation. The loss objective can be written as:

$$\mathcal{L}_{cod} = ||sg(\boldsymbol{e}_{\hat{k}}) - \boldsymbol{h}_e|| + ||\boldsymbol{e}_{\hat{k}} - sg(\boldsymbol{h}_e)||, \tag{19}$$

which can avoid generating trivial solutions. In addition, we enhance the generalization robustness to enhance the stability of node states with respect to different environment vectors as a regularization. Here, we replace $VQ(\boldsymbol{h}_e)$ with the other $\boldsymbol{e}_k$ in the codebook and minimize the variance of node states:

$$\mathcal{L}_{sta} = -\sum_k \sum_i \sum_t \left\| \mathbf{z}_i^t - \tilde{\boldsymbol{z}}_i^{t,(k)} \right\|^2, \tag{20}$$

where $\tilde{\boldsymbol{z}}_i^{t,(k)}$ is generated by replacing $VQ(\boldsymbol{h}_e)$ with $\boldsymbol{e}_k$. In summary, the whole loss objective can be summarized as:

$$\mathcal{L} = \mathcal{L}_{err} + \mathcal{L}_{cod} + \mathcal{L}_{sta} + \mathcal{L}_{mi}. \tag{21}$$

The whole algorithm of the proposed DGODE is summarized in Algorithm B.

Table 1. Details include number of nodes ($\#N$), variables ($\#V$), and total environments ($\#E$), with specifics on both seen and unseen environments for each benchmark.

BENCHMARKS	$\#N$	$\#V$	$\#E$	SEEN $\#E$	UNSEEN $\#E$
PROMETHEUS-T	15360	2	30	$\{a_1, a_2, ..., a_{25}\}$	$\{a_{26}, a_{27}, ..., a_{30}\}$
PROMETHEUS-P	22225	4	25	$\{b_1, b_2, ..., b_{20}\}$	$\{b_{21}, b_{22}, ..., b_{25}\}$
WEATHERBENCH-T	2048	1	3	$\{c_1, c_2\}$	$\{c_3\}$
WEATHERBENCH-W	2048	1	3	$\{d_1, d_2\}$	$\{d_3\}$
NAVIER–STOKES	4096	1	3	$\{e_1, e_2\}$	$\{e_3\}$

4. Experiments

4.1. Experimental Settings

In this section, we evaluate DGODE across benchmarks in combustion dynamics, meteorology, and PDEs, categorized as seen and unseen in Table 1. The model is trained in seen environments and tested in unseen ones to evaluate generalization ability.

Benchmarks. The study presents five benchmarks, as shown in Table 5, featuring two of our datasets: *Prometheus-T* for tunnel fire analysis (temperature and flue gas concentration) and *Prometheus-P* for pool fire analysis (velocity components, combined speed, and temperature), including 30 tunnel and 25 pool fire cases (details in Subsection 2.1). It also includes the *Weatherbench* dataset with two benchmarks: *Weatherbench-T* (temperature) and *Weatherbench-W* (wind speed) in various environments. The study additionally employs the *Navier-Stokes equation*, using the method from (Li et al., 2021), for training in two environments. Further details on environments, benchmark scales, and data resolution are in Appendix D.

Baselines. We evaluate DGODE against 13 notable models across three benchmarks, dividing them into four categories: ▷ **Visual Backbone Networks.** like U-Net (Ronneberger et al., 2015), Swin Transformer (Liu et al., 2021b), and Earthfarseer (Wu et al., 2023b); ▷ **Graph Neural Networks for spatio-temporal modeling** including CLCRN (Lin et al., 2022), MGNT (Pfaff et al., 2021), EAGLE (Janny et al., 2023), and DGCRN (Weng et al., 2023); ▷ **Graph-ODE series** with MPNODE (Gupta et al., 2022), CG-ODE (Huang et al., 2021) and SGODE (Chen et al., 2024a); ▷ **Operator learning methods** with FNO (Li et al., 2021), F-FNO (Tran et al., 2023), and LSM (Wu et al., 2023a). More details of baselines can be found in Appendix D.

Implementation & Evaluation Metrics. To ensure fairness, all methods train on a single NVIDIA-A100 using the ADAM optimizer for MSE loss over 500 epochs, with an initial learning rate of 10^{-3}. We set the batch size to 20. We study Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM) in our research. To adapt these models for irregular grids, we incorporate geo-FNO (Li et al., 2022b) for input/output transformation, facilitating the conversion of irregular domains into regular grids.

Table 2. In all benchmark tests, we compare the performance of our study with 12 baseline models and record the Mean Squared Error (MSE) as a performance metric. It's important to note that a lower MSE value indicates better model performance. Moreover, we define improvement as the reduction in relative error compared to the second-best model in each benchmark test.

Model	Benchmarks									
	Prometheus-T		Prometheus-P		WeatherBench-T		WeatherBench-W		Navier–Stokes	
	w/o OOD	*w/* OOD	*w/o* OOD	*w/* OOD	*w/o* OOD	*w/* OOD	*w/o* OOD	*w/* OOD	*w/o* OOD	*w/* OOD
U-Net (2015)	0.0931	0.1067	0.0471	0.0534	0.1382	0.1577	0.1650	0.1862	0.1982	0.2243
SwinT (2021b)	0.0652	0.0729	0.0564	0.0647	0.1191	0.1361	0.1420	0.1618	0.2248	0.2554
Earth (2023b)	0.0419	0.0470	0.0577	0.0654	0.0861	0.0952	0.1218	0.1374	0.1779	0.2013
CLCRN (2022)	0.0763	0.0851	0.1421	0.1587	0.1293	0.1471	0.1520	0.1710	0.3398	0.3873
MGNT (2021)	0.0967	0.1079	0.1672	0.1855	0.1331	0.1468	0.1591	0.1816	0.3561	0.3938
EAGLE (2023)	0.1128	0.1296	0.1981	0.2222	0.2781	0.3154	0.3300	0.3694	0.4731	0.5238
DGCRN (2023)	0.1027	0.1151	0.0961	0.1066	0.1587	0.1784	0.1845	0.2069	0.2198	0.2497
MPNODE (2022)	0.0652	0.0749	0.0897	0.0999	0.1665	0.1895	0.1941	0.2167	0.1991	0.2199
CG-ODE (2021)	0.0761	0.0843	0.0499	0.0565	0.0972	0.1107	0.1159	0.1320	0.2035	0.2243
SGODE (2024a)	0.0643	0.0723	0.0464	0.0537	0.0873	0.0931	0.1047	0.1301	0.1987	0.2109
FNO (2021)	0.0447	0.0506	0.0471	0.0522	0.1128	0.1290	0.1301	0.1466	0.1556	0.1712
F-FNO (2023)	0.0531	0.0608	0.1542	0.1728	0.1733	0.1934	0.2322	0.2556	0.2322	0.2664
LSM (2023a)	0.0414	0.0456	0.0489	0.0546	0.1298	0.1474	0.1549	0.1723	0.1535	0.1731
DGODE	**0.0344**	**0.0359**	**0.0397**	**0.0413**	**0.0843**	**0.0883**	**0.0976**	**0.1087**	**0.0805**	**0.0925**
Promotion	16.91%	21.27%	15.71%	20.88%	2.09%	7.25%	15.79%	17.65%	47.56%	46.56%

4.2. Performance Comparison under Environmental Distribution Shifts

In this section, we focus on discussing and evaluating the performance of DGODE and baseline models in predicting dynamic systems across various environmental distributions.

Results. The performance of compared approaches in different settings are recorded in Table 2. From the results, we can find that OOD shifts significantly impact model performance. All models, including U-Net and DGODE, show performance drops under distribution shifts. For example, in the Prometheus-T test, U-Net's MSE increased from 0.0931 to 0.1067, while DGODE's rose from 0.0344 to 0.0359. This pattern is consistent across models, underscoring challenges in adapting to new data distributions. However, DGODE remains robust against these shifts, maintaining lower MSE scores than others, such as in the WeatherBench-T test where DGODE's MSE was 0.0883 (*w/* OOD) compared to the Earth model. DGODE's performance consistency is due to its generalization abilities.

Visualization Case Study. Figure 3 provides a clear comparison of different methods, highlighting DGODE's effectiveness in various fields like meteorological forecasting and physical dynamics simulation. DGODE's accuracy is evident in flame flow field predictions, closely matching ground truth for temperature and velocity, and in time series prediction error figures. Its application to Navier-Stokes equations, crucial for fluid motion, showcases its capability to accurately replicate complex fluid dynamics. Furthermore, DGODE demonstrates robustness in diverse scenarios, including turbulence modeling and atmospheric data assimilation, maintaining high precision across different scales and conditions. This versatility underscores its potential for broad application in scientific and engineering problems. For additional results, refer to Appendix F.

4.3. Model Analysis

We analyze DGODE, focusing on how different environmental conditions and prediction lengths affect model performance. We compare the Prometheus-P and WeatherBench-W to assess performance across various settings, including unseen environments $\{b_{21}, b_{22}\}$ and $\{d_3\}$. We explore environmental embeddings learning, visualize Codebank vectors using PCA, and conduct sensitivity analysis on key hyperparameters. Finally, we evaluate crucial model components through ablation studies and discuss their performance in transfer learning.

Effect of Different Environment and Prediction Lengths. Our study focuses on the Prometheus-P and WeatherBench-W cases, examining prediction performance in different environments and over various forecast lengths. We test our model in previously unseen environments $\{b_{21}, b_{22}\}$ and $\{d_3\}$. For Prometheus-P, forecast lengths of $\{20, 40, 60, 80, 100, 120\}$ steps are analyzed, while for WeatherBench-W, we use lengths of $\{2, 4, 6, 8, 10, 12\}$. As illustrated in Figure 4, our DGODE model outperforms others in all test scenarios. Particularly, DGODE maintains high local fidelity even at 120 prediction steps, in contrast to the LSM model, which becomes overly smooth. DGODE also achieves higher SSIM scores, highlighting its strength in OOD and long-term forecasts.

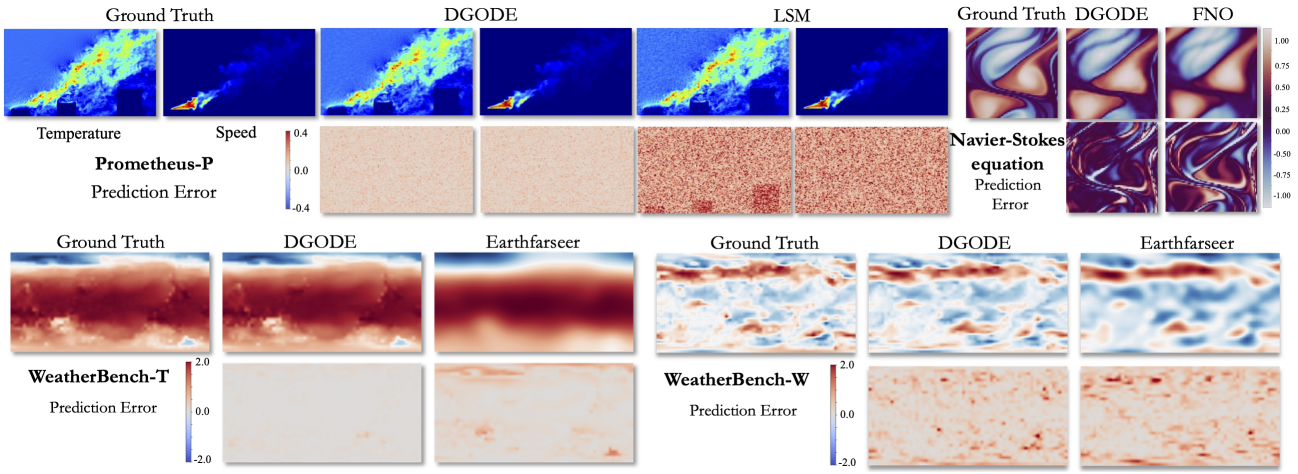


Figure 3. TOP: Prometheus-P shows the visualization of temperature and velocity variables at the last timestamp (T=50). Navier-Stokes equation prediction shows results at the last timestamp (T=10). *Bottom*: For WeatherBench-T and WeatherBench-W, we display the prediction visualization at the last timestamp of the time series (T=12). To more clearly demonstrate the accuracy of the predictions, we also created images of prediction errors (Error $= Y - \hat{Y}$). More detailed case analyses and examples are available in Appendix F.

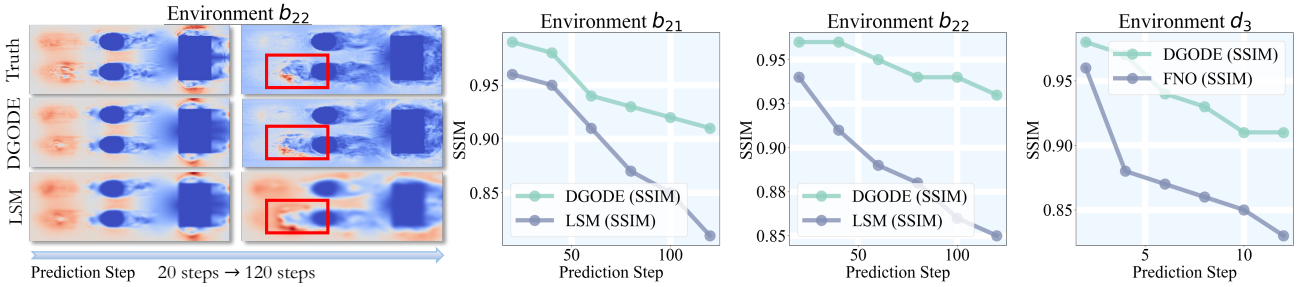
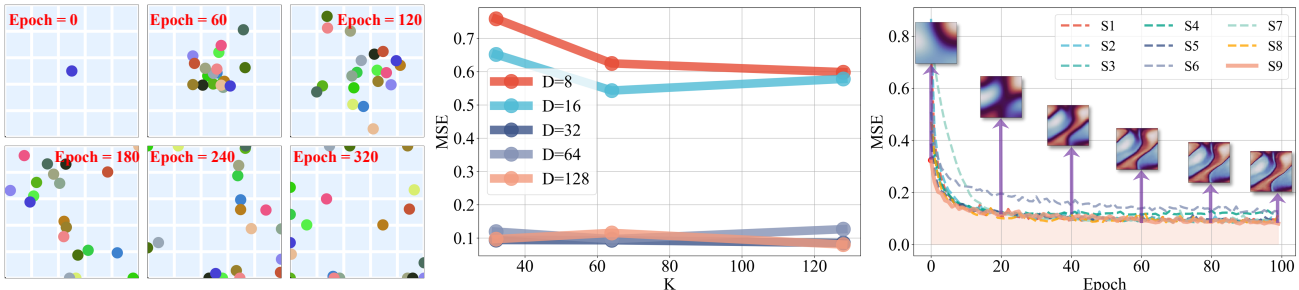


Figure 4. Left shows a top view of the Prometheus-P dataset, comparing the DGODE model with the runner-up LSM model. It displays the prediction results at 20 and 120 steps. *Right*, the changes in SSIM over time steps are shown for the DGODE, LSM, and FNO models on both Prometheus-P and WeatherBench-W datasets.

Learning of Environmental Embeddings. In our study, we employ Principal Component Analysis (PCA) as a dimensionality reduction technique to visualize the environmental Codebank vectors in the Prometheus-P dataset. The visualization results are shown in Figure 5 (a). Our aim is to explore the effectiveness of the environmental Codebank as a potential feature representation. During this process, we focus on 20 different colored environmental embeddings. These embeddings represent unique environments in the physical world. They start from the same initial point and gradually disperse in multiple directions as the training cycles progress. This phenomenon indicates that our model effectively learns information-rich environmental representations.

Sensitivity Analysis. Our study examines the influence of key Codebank hyperparameters, K (Codebank size) and D (dimension of latent vectors), on DGODE's performance in Navier-Stokes simulations. We varied K among $\{8, 16, 32, 64, 128\}$ and D among $\{32, 64, 128\}$, with results in Figure 5 (b) showing that performance improves with larger D, as smaller dimensions are insufficient for learning the environment. Figure 5 (c) reveals that irrespective of initial differences, models with various D and K values converge to similar performance. We also depict the model's learning progression in the best training scenario, demonstrating its increasing proficiency in capturing complex patterns in Navier-Stokes equations.

Transferability. Prometheus-P and Prometheus-T come from FDS modeling, but they have different scenarios. As shown in Table 3, to evaluate the DGODE model transferability, we fine-tune the model trained on Prometheus-P for different scenes. This also demonstrates the model's strong ability to handle unknown data (OOD). It is important to note that our proposed DGODE always shows good forward transfer abilities, even with limited data. Moreover, DGODE achieves the best performance, whether using pretrained models or starting from scratch. This indicates that DGODE can extract key physical features from complex dynamics system.



(a) Visualization of Environment embeddings (b) Effects of ***K*** and ***D*** on Navier-Stokes (c) Training progress with different ***K*** and ***D***

Figure 5. Figure (a) shows the visualization of environment embeddings at different training epochs. Figure (b) describes the impact of hyperparameters on the model in Navier-Stokes equation (The viscosity coefficient is 10^{-5}). Figure (c) visualizes the training process.

Table 3. Transfer the model pre-trained from full-data Prometheus-P to limited-data Prometheus-T. The results are presented in the formalization of $P \rightarrow T$, where P is the model performance when it is trained from scratch and T is the performance finetuned from the Prometheus-P pre-trained model.

$\text{MSE}_{\times 10^2}$	20% PROMETHEUS-T	40% PROMETHEUS-T	60% PROMETHEUS-T	80% PROMETHEUS-T	100% PROMETHEUS-T
EARTH	5.31→5.28 (+0.56%)	5.22→5.12 (+1.92%)	5.01→4.66 (+6.99%)	4.89→4.81 (+1.64%)	4.70→4.54 (+3.40%)
CG-ODE	10.12→10.97 (-8.40%)	9.98→9.63 (+3.51%)	9.15→9.01 (+1.53%)	8.92→8.19 (+8.18%)	8.43→8.97 (-6.41%)
LSM	8.02→7.97 (+0.62%)	7.98→6.33 (+20.68%)	6.42→6.07 (+5.45%)	5.12→5.09 (+0.59%)	4.56→4.47 (+1.97%)
DGODE	**5.91→5.62** (+4.91%)	**4.23→4.18** (+1.18%)	**3.97→3.82** (+3.78%)	**3.87→3.66** (+5.43%)	**3.59→3.51** (+2.23%)

Table 4. Ablation Studies on two benchmark.

VARIANTS	BENCHMARKS			
	PROMETHEUS-T		WEATHERBENCH-T	
	w/o OOD	*w/* OOD	*w/o* OOD	*w/* OOD
DGODE w/o $\mathcal{L}_{cod}$	0.0354	0.0367	0.0867	0.0901
DGODE w/o $\mathcal{L}_{sta}$	0.0357	0.0418	0.0899	0.0957
DGODE w/o $\mathcal{L}_{mi}$	0.0367	0.0421	0.0865	0.0973
DGODE	0.0344	0.0359	0.0843	0.0883

Ablation Study. We conduct extensive ablation experiments to demonstrate the performance of the key components in DGODE, as shown in Table 4. To thoroughly understand the impact of each component, we introduce the following model variants: (1) DGODE w/o $\mathcal{L}_{cod}$ removes the codebook loss, which is designed to improve the representation quality and efficiency; (2) DGODE w/o $\mathcal{L}_{sta}$ removes the stability loss, which contributes to the stability of the model by preventing overfitting and enhancing generalization; (3) DGODE w/o $\mathcal{L}_{mi}$ removes the mutual information minimization loss, aimed at reducing redundant information and improving the robustness of the model. The experiment shows that taking out any part lowers performance on all benchmarks. For example, removing $\mathcal{L}_{mi}$ increases the MSE by about 22% on the Prometheus-T dataset and by about 15% on the Weatherbench-T dataset. This demonstrates that mutual information minimization loss is crucial for maintaining robustness and accuracy in OOD scenarios. Moreover, our DGODE outperforms DGODE w/o $\mathcal{L}_{sta}$, which validates that minimizing the variance as a regularization can improve the model performance.

5. Conclusion

In this paper, we study the problem of out-of-distribution fluid dynamics modeling and propose a new large-scale dataset *Prometheus*, which simulates tunnel and pool fires across different environments. We also construct an extensive benchmark with 12 baselines and propose a new approach named DGODE for this problem. Our DGODE learns disentangled node features and environment features from historical trajectories using mutual information minimization. These features would be incorporated into an environment-aware graph ODE framework, where the latent states of nodes and the environment are modeled simultaneously. Extensive experiments validate the superiority of the proposed DGODE compared with a variety of competing baselines. Our Prometheus dataset can be found at the following link: https://huggingface.co/datasets/easylearning/Prometheus.

Impact Statement

This paper constructs a new large-scale dataset *Prometheus* as well as an extensive benchmark for out-of-distribution fluid dynamics modeling, which can benefit the research on scientific machine learning. Moreover, we build an effective data-driven approach DGODE for this problem, which can be applied to understand complicated interacting dynamics in fluid mechanics. In future works, we would extend our proposed DGODE to different scientific scenarios such as rigid dynamics and molecular dynamics.

References

Baradel, F., Neverova, N., Mille, J., Mori, G., and Wolf, C. Cophy: Counterfactual learning of physical dynamics. *arXiv preprint arXiv:1909.12000*, 2019.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. Mutual information neural estimation. In *International conference on machine learning*, pp. 531–540. PMLR, 2018.

Chen, L., Wu, K., Lou, J., and Liu, J. Signed graph neural ordinary differential equation for modeling continuous-time dynamics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8292–8301, 2024a.

Chen, P., Zhang, Y., Cheng, Y., Shu, Y., Wang, Y., Wen, Q., Yang, B., and Guo, C. Multi-scale transformers with adaptive pathways for time series forecasting. In *ICLR*, 2024b.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *NeurIPS*, 2018.

Deng, A. and Hooi, B. Graph neural network-based anomaly detection in multivariate time series. *AAAI*, 2021.

Deng, X., Wang, W., Feng, F., Zhang, H., He, X., and Liao, Y. Counterfactual active learning for out-of-distribution generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11362–11377, 2023.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. In *NeurIPS*, 2019.

Fang, Z. A high-efficient hybrid physics-informed neural networks based on convolutional neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5514–5526, 2021.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. How to train your neural ode: the world of jacobian and kinetic regularization. In *ICML*, pp. 3154–3164, 2020.

Gui, S., Liu, M., Li, X., Luo, Y., and Ji, S. Joint learning of label and environment causal independence for graph out-of-distribution generalization. *arXiv preprint arXiv:2306.01103*, 2023.

Guo, X., Li, W., and Iorio, F. Convolutional neural networks for steady flow approximation. In *KDD*, pp. 481–490, 2016.

Gupta, J. K., Vemprala, S., and Kapoor, A. Learning modular simulations for homogeneous systems. In *NeurIPS*, 2022.

Han, J., Huang, W., Ma, H., Li, J., Tenenbaum, J. B., and Gan, C. Learning physical dynamics with subequivariant graph neural networks. *arXiv preprint arXiv:2210.06876*, 2022a.

Han, X., Gao, H., Pffaf, T., Wang, J.-X., and Liu, L.-P. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*, 2022b.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021.

Huang, C., Zhao, L., Niu, J., Di, J., Yuan, J., Zhao, Q., Zhang, F., Zhang, Z., Lei, J., and He, G. Coupled particle and mesh method in an euler frame for unsteady flows around the pitching airfoil. *Engineering Analysis with Boundary Elements*, 138:159–176, 2022.

Huang, X., Shi, W., Meng, Q., Wang, Y., Gao, X., Zhang, J., and Liu, T.-Y. Neuralstagger: accelerating physics-constrained neural pde solver with spatial-temporal decomposition. In *ICML*, 2023a.

Huang, Z., Sun, Y., and Wang, W. Learning continuous system dynamics from irregularly-sampled partial observations. In *NeurIPS*, pp. 16177–16187, 2020.

Huang, Z., Sun, Y., and Wang, W. Coupled graph ode for learning interacting system dynamics. In *KDD*, 2021.

Huang, Z., Sun, Y., and Wang, W. Generalizing graph ode for learning complex system dynamics across environments. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 798–809, 2023b.

Janny, S., Beneteau, A., Thome, N., Nadri, M., Digne, J., and Wolf, C. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers. *arXiv preprint arXiv:2302.10803*, 2023.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

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

Kostic, V. R., Novelli, P., Grazzi, R., Lounici, K., and Pontil, M. Learning invariant representations of time-homogeneous stochastic dynamical systems. In *ICLR*, 2024.

Le Clainche, S., Ferrer, E., Gibson, S., Cross, E., Parente, A., and Vinuesa, R. Improving aircraft performance using machine learning: a review. *Aerospace Science and Technology*, pp. 108354, 2023.

Li, H., Pan, S. J., Wang, S., and Kot, A. C. Domain generalization with adversarial feature learning. In *CVPR*, pp. 5400–5409, 2018.

Li, H., Wang, Y., Wan, R., Wang, S., Li, T.-Q., and Kot, A. Domain generalization for medical imaging classification with linear-dependency regularization. In *NeurIPS*, pp. 3118–3129, 2020.

Li, J., Song, Z., and Yang, B. Nvfi: Neural velocity fields for 3d physics learning from dynamic videos. In *NeurIPS*, 2023a.

Li, X., Dai, Y., Ge, Y., Liu, J., Shan, Y., and Duan, L.-Y. Uncertainty modeling for out-of-distribution generalization. *arXiv preprint arXiv:2202.03958*, 2022a.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2021.

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022b.

Li, Z., Kovachki, N. B., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., et al. Geometry-informed neural operator for large-scale 3d pdes. In *NeurIPS*, 2023b.

Lienen, M., Lüdke, D., Hansen-Palmus, J., and Günnemann, S. From zero to turbulence: Generative modeling for 3d flow simulation. In *ICLR*, 2024.

Lin, H., Gao, Z., Xu, Y., Wu, L., Li, L., and Li, S. Z. Conditional local convolution for spatio-temporal meteorological forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7470–7478, 2022.

Lippe, P., Veeling, B. S., Perdikaris, P., Turner, R. E., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. In *NeurIPS*, 2023.

Liu, J., Shen, Z., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021a.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. C.-F., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021b.

Lu, W., Wang, J., Wang, Y., Ren, K., Chen, Y., and Xie, X. Towards optimization and model selection for domain generalization: A mixup-guided solution. *arXiv preprint arXiv:2209.00652*, 2022.

Luo, X., Gu, Y., Jiang, H., Huang, J., Ju, W., Zhang, M., and Sun, Y. Graph ode with factorized prototypes for modeling complicated interacting dynamics. *arXiv preprint arXiv:2311.06554*, 2023a.

Luo, X., Yuan, J., Huang, Z., Jiang, H., Qin, Y., Ju, W., Zhang, M., and Sun, Y. Hope: High-order graph ode for modeling interacting dynamics. In *International Conference on Machine Learning*, pp. 23124–23139. PMLR, 2023b.

Luo, X., Zhao, Y., Mao, Z., Qin, Y., Ju, W., Zhang, M., and Sun, Y. Rignn: A rationale perspective for semi-supervised open-world graph classification. *Transactions on Machine Learning Research*, 2023c.

Luo, X., Wang, H., Huang, Z., Jiang, H., Gangan, A., Jiang, S., and Sun, Y. Care: Modeling interacting dynamics under temporal environmental variation. *Advances in Neural Information Processing Systems*, 36, 2024.

Ma, P., Chen, P. Y., Deng, B., Tenenbaum, J. B., Du, T., Gan, C., and Matusik, W. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *ICML*, 2023.

Obiols-Sales, O., Vishnu, A., Malaya, N., and Chandramowliswharan, A. Cfdnet: A deep learning-based accelerator for fluid simulations. In *Proceedings of the 34th ACM international conference on supercomputing*, pp. 1–12, 2020.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. In *ICLR*, 2021.

Rämä, M. and Sipilä, K. Transition to low temperature distribution in existing systems. *Energy Procedia*, 116: 58–68, 2017.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *ICML*, pp. 8459–8468, 2020.

Shao, Y., Loy, C. C., and Dai, B. Transformer with implicit edges for particle-based physics simulation. In *ECCV*, pp. 549–564, 2022.

Steeven, J., Madiha, N., Julie, D., and Christian, W. Space and time continuous physics simulation from partial observations. In *ICLR*, 2024.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. In *ICLR*, 2023.

Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Volpi, R., Namkoong, H., Sener, O., Duchi, J. C., Murino, V., and Savarese, S. Generalizing to unseen domains via adversarial data augmentation. In *NeurIPS*, 2018.

Wang, K., Wu, H., Duan, Y., Zhang, G., Wang, K., Peng, X., Zheng, Y., Liang, Y., and Wang, Y. Nuwadynamics: Discovering and updating in causal spatio-temporal modeling.

Wang, K., Liang, Y., Li, X., Li, G., Ghanem, B., Zimmermann, R., Yi, H., Zhang, Y., Wang, Y., et al. Brave the wind and the waves: Discovering robust and generalizable graph lottery tickets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023a.

Wang, K., Wu, H., Zhang, G., Fang, J., Liang, Y., Wu, Y., Zimmermann, R., and Wang, Y. Modeling spatio-temporal dynamical systems with neural discrete learning and levels-of-experts. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Wang, R., Mao, W., and Li, H. Deepsimho: Stable pose estimation for hand-object interaction via physics simulation. In *NeurIPS*, 2023b.

Wang, Y., Li, X., Qi, Z., Li, J., Li, X., Meng, X., and Meng, L. Meta-causal feature learning for out-of-distribution generalization. In *ECCV*, pp. 530–545, 2022.

Wang, Z., Loog, M., and van Gemert, J. Respecting domain relations: Hypothesis invariance for domain generalization. In *ICPR*, pp. 9756–9763, 2021.

Weng, W., Fan, J., Wu, H., Hu, Y., Tian, H., Zhu, F., and Wu, J. A decomposition dynamic graph convolutional recurrent network for traffic forecasting. *Pattern Recognition*, 142:109670, 2023.

Wenzel, F., Dittadi, A., Gehler, P., Simon-Gabriel, C.-J., Horn, M., Zietlow, D., Kernert, D., Russell, C., Brox, T., Schiele, B., et al. Assaying out-of-distribution generalization in transfer learning. In *NeurIPS*, pp. 7181–7198, 2022.

Wu, H., Zhou, S., Huang, X., and Xiong, W. Neural manifold operators for learning the evolution of physical dynamics.

Wu, H., Hu, T., Luo, H., Wang, J., and Long, M. Solving high-dimensional pdes with latent spectral models. *arXiv preprint arXiv:2301.12664*, 2023a.

Wu, H., Wang, S., Liang, Y., Zhou, Z., Huang, W., Xiong, W., and Wang, K. Earthfarseer: Versatile spatio-temporal dynamical systems modeling in one model. *AAAI2024*, 2023b.

Wu, H., Xion, W., Xu, F., Luo, X., Chen, C., Hua, X.-S., and Wang, H. Pastnet: Introducing physical inductive biases for spatio-temporal video prediction. *arXiv preprint arXiv:2305.11421*, 2023c.

Wu, L., Qiu, X., Yuan, Y.-x., and Wu, H. Parameter estimation and variable selection for big systems of linear ordinary differential equations: A matrix-based approach. *Journal of the American Statistical Association*, 2018.

Wu, Q., Zhang, H., Yan, J., and Wipf, D. Handling distribution shifts on graphs: An invariance perspective. *arXiv preprint arXiv:2202.02466*, 2022.

Xhonneux, L.-P., Qu, M., and Tang, J. Continuous graph neural networks. In *ICML*, pp. 10432–10441, 2020.

Xia, Y., Liang, Y., Wen, H., Liu, X., Wang, K., Zhou, Z., and Zimmermann, R. Deciphering spatio-temporal graph forecasting: A causal lens and treatment. *Advances in Neural Information Processing Systems*, 36, 2024.

Xiong, W., Xiang, Y., Wu, H., Zhou, S., Sun, Y., Ma, M., and Huang, X. Ai-goms: Large ai-driven global ocean modeling system. *arXiv preprint arXiv:2308.03152*, 2023.

Yang, C., Wu, Q., Wen, Q., Zhou, Z., Sun, L., and Yan, J. Towards out-of-distribution sequential event prediction: A causal treatment. *arXiv preprint arXiv:2210.13005*, 2022.

Yu, H.-X., Zheng, Y., Gao, Y., Deng, Y., Zhu, B., and Wu, J. Inferring hybrid neural fluid fields from videos. In *NeurIPS*, 2023.

Yu, Y.-Y., Choi, J., Cho, W., Lee, K., Kim, N., Chang, K., Woo, C., Kim, I., Lee, S., Yang, J. Y., et al. Learning flexible body collision dynamics with hierarchical contact mesh transformer. In *ICLR*, 2024.

Zang, C. and Wang, F. Neural dynamics on complex networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 892–902, 2020.

Zhan, X., Dai, Z., Wang, Q., Li, Q., Xiong, H., Dou, D., and Chan, A. B. Pareto optimization for active learning under out-of-distribution data scenarios. *Transactions on Machine Learning Research*, 2023.

Zhang, G., Yue, Y., Wang, K., Fang, J., Sui, Y., Wang, K., Liang, Y., Cheng, D., Pan, S., and Chen, T. Two heads are better than one: Boosting graph sparse training via semantic and topological awareness, 2024.

Zhang, Y., Gao, S., Pei, J., and Huang, H. Improving social network embedding via new second-order continuous graph neural networks. In *KDD*, pp. 2515–2523, 2022.

Zhao, X., Meng, L., Tong, X., Xu, X., Wang, W., Miao, Z., Mo, D., et al. A novel computational fluid dynamic method and validation for assessing distal cerebrovascular microcirculatory resistance. *Computer Methods and Programs in Biomedicine*, 230:107338, 2023.

Zhou, D., Wang, K., Gu, J., Peng, X., Lian, D., Zhang, Y., You, Y., and Feng, J. Dataset quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17205–17216, 2023.

A. Proof of Theorem 3.1.

Proof of Theorem 3.1. We directly consider the case where the time points tend to infinity; that is

$$\frac{1}{n}\sum_{j=1}^{n}\|\tilde{\boldsymbol{z}}(t_j)-e^{t_j\boldsymbol{W}_1}\boldsymbol{h}\|_2^2\to\int_0^1\|\tilde{\boldsymbol{z}}(t)-e^{t\boldsymbol{W}_1}\boldsymbol{h}\|_2^2dt\equiv\mathcal{L}(\boldsymbol{W}_1).$$

Define $\boldsymbol{E}_1=(\boldsymbol{I}_p,\boldsymbol{0}_p)\in\mathbb{R}^{p\times(2p)}$ be the extractor matrix such that $\boldsymbol{E}_1\boldsymbol{A}$ returns the first p rows of the matrix $\boldsymbol{A}$. Similarly, we define $\boldsymbol{E}_1=(\boldsymbol{0}_p,\boldsymbol{I}_p)$. By Eqn. 14–Eqn. 15,

$$\mathcal{L}(\boldsymbol{W}_1)=p\sigma^2+\int_0^1\|\boldsymbol{E}_1e^{t\boldsymbol{W}^*}\boldsymbol{Z}(0)-e^{t\boldsymbol{W}_1}\boldsymbol{h}\|_2^2dt.$$

Define $\boldsymbol{D}(\boldsymbol{W},t)$ as the following $p\times p^2$-dimensional matrix function

$$\boldsymbol{D}_{i,(\ell-1)p+k}(\boldsymbol{W},t)=\frac{\partial\big(e^{t\boldsymbol{W}}\boldsymbol{h}\big)_i}{\partial W_{k\ell}},\quad i,\ell,k=1,\ldots,p,\tag{22}$$

where $W_{k\ell}$ denotes the (k,ℓ)th entry of $\boldsymbol{W}$. Note that the second order derivatives of $\mathcal{L}(\boldsymbol{W}_1)$ are

$$\frac{\partial^2\mathcal{L}(\boldsymbol{W}_1)}{\partial W_{k\ell}\partial W_{k'\ell'}}=\int_0^1\sum_{i=1}^{p}\frac{\partial\big(e^{t\boldsymbol{W}_1}\boldsymbol{h}\big)_i}{\partial W_{k\ell}}\frac{\partial\big(e^{t\boldsymbol{W}_1}\boldsymbol{h}\big)_i}{\partial W_{k'\ell'}}dt.$$

Thus, the Hessian matrix of $\mathcal{L}(\boldsymbol{W}_1)$ (after $\boldsymbol{W}_1$ being vectorized) is $\int_0^1\{\boldsymbol{D}(\boldsymbol{W}_1,t)\}^T\boldsymbol{D}(\boldsymbol{W}_1,t)dt$, which is a positive semidefinite matrix. This implies that the global minimizer $\widehat{\boldsymbol{W}}_1$ of $\mathcal{L}(\boldsymbol{W}_1)$ must be the first-order stationary point; that is,

$$\nabla\mathcal{L}(\widehat{\boldsymbol{W}}_1)=2\int_0^1\{\boldsymbol{D}(\widehat{\boldsymbol{W}}_1,t)\}^T(e^{t\widehat{\boldsymbol{W}}_1}\boldsymbol{h}-\boldsymbol{E}_1e^{t\boldsymbol{W}^*}\boldsymbol{Z}(0))dt=\boldsymbol{0}.$$

By Lemma S6.3 in Wu et al. (2018), we have

$$\boldsymbol{D}(\boldsymbol{W},t)=t\int_0^1(\boldsymbol{h}^Te^{ts\boldsymbol{W}^T})\otimes e^{t(1-s)\boldsymbol{W}}ds,$$

which gives

$$\nabla\mathcal{L}(\widehat{\boldsymbol{W}}_1)=2\int_0^1\int_0^1 t(e^{ts\widehat{\boldsymbol{W}}_1}\boldsymbol{h})\otimes e^{t(1-s)\widehat{\boldsymbol{W}}_1^T}\big(e^{t\widehat{\boldsymbol{W}}_1}\boldsymbol{h}-\boldsymbol{E}_1e^{t\boldsymbol{W}^*}\boldsymbol{Z}(0)\big)dsdt=\boldsymbol{0}.$$

In general, it is difficult to obtain an explicit form of $\widehat{\boldsymbol{W}}_1$. But we can show that $\boldsymbol{W}_1^*$ is not a zero of $\nabla\mathcal{L}(\boldsymbol{W}_1)$ when each entry of $\boldsymbol{W}^*$ and $\boldsymbol{Z}(0)$ is strictly positive. To see this, first observe that for any matrix $\boldsymbol{A}$,

$$e^{t\boldsymbol{A}}=\boldsymbol{I}+\sum_{k=1}^{\infty}\frac{t^k}{k!}\boldsymbol{A}^k,$$

and thus

$$e^{t\boldsymbol{W}_1^*}\boldsymbol{h}-\boldsymbol{E}_1e^{t\boldsymbol{W}^*}\boldsymbol{Z}(0)=\sum_{k=1}^{\infty}\frac{t^k}{k!}\big(\boldsymbol{W}_1^{*k}\boldsymbol{h}-\boldsymbol{E}_1e^{t\boldsymbol{W}^*}\boldsymbol{E}_1^T\boldsymbol{h}-\boldsymbol{E}_1e^{t\boldsymbol{W}^*}\boldsymbol{E}_2^T\boldsymbol{h}_e\big).$$

Moreover, since $\boldsymbol{W}^*=\begin{pmatrix}\boldsymbol{W}_1^*,\boldsymbol{W}_2^*\\ \boldsymbol{W}_3^*,\boldsymbol{W}_4^*\end{pmatrix}$, by the positivity of entries of $\boldsymbol{W}^*$ we have that each entry of $\boldsymbol{W}_1^{*k}-\boldsymbol{E}_1\boldsymbol{W}^{*k}\boldsymbol{E}_1^T$ and $-\boldsymbol{E}_1\boldsymbol{W}^{*k}\boldsymbol{E}_2^T$ is strictly negative. It follows immediately from the positivity of components of $\boldsymbol{Z}(0)$ that each components of $e^{t\widehat{\boldsymbol{W}}_1}\boldsymbol{h}-\boldsymbol{E}_1e^{t\boldsymbol{W}^*}\boldsymbol{Z}(0)$ is strictly negative. As the entries of $t(e^{ts\widehat{\boldsymbol{W}}_1}\boldsymbol{h})\otimes e^{t(1-s)\widehat{\boldsymbol{W}}_1^T}$ are all strictly positive, we conclude that each component of $\nabla\mathcal{L}(\boldsymbol{W}_1^*)$ are strictly negative. □

B. Algorithm

We summarize the overall framework of DGODE in Algorithm 1.

Algorithm 1 DGODE Framework for OOD Fluid Dynamics Modeling

Require: Historical trajectories X^H, Environment E
Ensure: Future trajectories X^F
Initialize Temporal GNN and Frequency Network
Initialize Vector Quantization (VQ) Codebook with K environment vectors $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_K\}$
{**Feature Extraction**}
for each node i in the graph G **do**
  Extract spatial node features $\boldsymbol{h}_i^{t,(l)}$ using Temporal GNN
  Extract frequency-based node features $\boldsymbol{h}_i^{(f)}$ using Frequency network
  Concatenate spatial and frequency-based features to form final node features $\boldsymbol{h}_i$
end for
for each environment **do**
  Extract environment features $\boldsymbol{h}_e$ using a separate backbone network
  Summarize all node features in the environment to form environment representation
end for
{**Representation Disentanglement**}
Disentangle node and environment features using mutual information minimization:
Apply Vector Quantization (VQ) to each environment feature $\boldsymbol{h}_e$
Minimize mutual information between node features $\boldsymbol{h}_i$ and quantized environment representations
{**Environment-aware Graph ODE**}
Initialize node states $\boldsymbol{z}_i^0$ and environment states $\boldsymbol{z}_e^0$ from extracted features
Model the interacting dynamics using environment-awre graph ODEs:
Update node states $\boldsymbol{z}_i$ over time using equation Eqn. 9
Update environment states $\boldsymbol{z}_e$ over time using equation Eqn. 10
{**Optimization**}
Define loss functions:
Reconstruction loss $\mathcal{L}_{err}$ for the prediction accuracy
Codebook loss $\mathcal{L}_{cod}$ for vector quantization
Stability loss $\mathcal{L}_{sta}$ for enhancing generalization robustness
Mutual information loss $\mathcal{L}_{mi}$ for representation disentanglement
Train the model by minimizing the total loss: $\mathcal{L} = \mathcal{L}_{err} + \mathcal{L}_{cod} + \mathcal{L}_{sta} + \mathcal{L}_{mi}$
Predict future trajectories X^F using the trained model
Return X^F

C. Related Work

C.1. Fluid Dynamics Modeling

Learning-based fluid dynamics modeling (Obiols-Sales et al., 2020; Lienen et al., 2024; Ma et al., 2023; Wu et al., 2023c; Xiong et al., 2023) has received extensive attention in the field of scientific machine learning (Kostic et al., 2024; Chen et al., 2024b; Wang et al., 2023b) with applications ranging from aerospace engineering (Le Clainche et al., 2023) to biomedicine (Zhao et al., 2023). Early efforts focus on leveraging convolutional neural networks (Fang, 2021; Guo et al., 2016) to learn from physical simulations with regular grids. Recent approaches (Pfaff et al., 2021; Shao et al., 2022; Wang et al., 2024) attempt to leverage geometric graphs for finer-level simulation using GNNs, which follow the message passing mechanism to update node representations iteratively. However, these approaches usually focus on next-time prediction (Pfaff et al., 2021; Shao et al., 2022) or auto-regressive prediction (Han et al., 2022b), which is hard to capture underlying continuous dynamics in fluid simulations (Lippe et al., 2023; Ma et al., 2023). Moreover, these approaches usually assume that training and test datasets come from the same distribution. In contrast, this work focuses on out-of-distribution fluid dynamics and proposes a new dataset as well as a benchmark to benefit research on this underexplored problem.

C.2. Neural Ordinary Differential Equations

Neural Ordinary Differential Equations (ODEs) (Chen et al., 2018) incorporate ODE into learning-based neural networks, which has received extensive interest in machine learning and computational mathematics. Neural ODEs have been widely to depict continuous-time dynamical systems with the benefit of learning from irregularly sampled observations (Huang et al., 2020; Zang & Wang, 2020). A range of approaches have been developed to enhance the capacity of neural ODEs such as including regularization terms (Finlay et al., 2020) and dimension augmentation (Dupont et al., 2019). Recently, neural ODEs have been combined with GNNs to solve overfitting issues (Xhonneux et al., 2020) and provide the explainability of models (Zhang et al., 2022). Furthermore, neural ODEs combined with GNNs have solved dynamical system prediction problems. For example, SGODE (Chen et al., 2024a) uses signed graph neural ordinary differential equations for continuous-time dynamics modeling, while HOPE (Luo et al., 2023b) employs second-order graph ODE functions to capture long-term dependencies in complex dynamic systems. In this work, we propose a neural ODE-based framework named DGODE to solve OOD fluid dynamics modeling, which models the state evolution of both node representations and the environment simultaneously.

C.3. Out-of-distribution Generalization

Out-of-distribution (OOD) generalization (Volpi et al., 2018; Wang et al., 2023a; Wu et al., 2022; Yang et al., 2022) aims to enhance the model performance when training and test data come from different distributions. This problem has been studied in various scenarios such as time-series forecasting (Zhang et al., 2024), image classification (Li et al., 2022a) and graph data mining (Gui et al., 2023). A wide range of approaches adopt invariant learning for OOD generalization (Li et al., 2020; Wang et al., 2021; Li et al., 2018), which generates domain-invariant features in the latent space. In this way, these can delete the influence of spurious correlations resulting from the distribution shift. In addition, causal inference (Wang et al., 2022; Wang et al.), model selection (Lu et al., 2022; Wenzel et al., 2022) and active learning (Zhan et al., 2023; Deng et al., 2023) have also been adopted to enhance OOD performance in practical scenarios. In this work, we study a practical yet underexplored problem of OOD fluid dynamics modeling and propose a new large-scale OOD dataset and an extensive benchmark to facilitate researchers studying this topic.

D. Experimental setup detials

Baselines. We evaluate DGODE against 12 notable models across three benchmarks, dividing them into four categories:

- **Vision backbone networks:** (1). U-Net (Ronneberger et al., 2015) is a deep learning architecture for medical image segmentation. It features a symmetric contracting path and an expansive path for precise localization of areas of interest in images. (2). Swin Transformer (Liu et al., 2021b) is a Transformer-based network architecture mainly used for computer vision tasks. It addresses the issue of varying image scales efficiently through a hierarchical Transformer structure. Earthfarseer (Wu et al., 2023b) is a deep learning model designed for Earth observation data. Its specific network structure enhances monitoring and prediction capabilities for environmental and meteorological phenomena.

- **Graph Neural Networks for spatio-temporal modeling:** (1). CLCRN (Lin et al., 2022) is a conditional graph convolutional recurrent neural network for spatio-temporal data processing. It effectively captures spatio-temporal dependencies, suitable for scenarios like traffic flow prediction. (2). MGNT (Pfaff et al., 2021) is a multi-graph neural network focusing on learning complex relationships from multiple graph structures. It excels in areas like physical system modeling. EAGLE (Janny et al., 2023) is a graph neural network for spatio-temporal data, emphasizing effective learning in dynamic environments. It is suitable for fields like weather prediction. DGCRN (Weng et al., 2023) is a decomposed graph convolutional recurrent network. Its decomposition approach enhances the processing of spatio-temporal data, especially effective in traffic and network flow prediction.

- **Graph-ODE series:** (1). MPNODE (Gupta et al., 2022) is a model combining graph neural networks and ordinary differential equations (ODE) for modeling dynamic graph data. It captures the temporal evolution characteristics of graph-structured data. (2). CG-ODE (Huang et al., 2021) is a coupled graph ordinary differential equation model. It processes graph structures and continuous time dynamics simultaneously, suitable for modeling complex network systems. (3).SGODE (Chen et al., 2024a) is a method that integrates graph neural networks with ordinary differential equations. It is especially effective for modeling complex system dynamics that involve positive and negative relationships.

Table 5. Details for benchmarks. The input-output shapes are presented in the shape of (temporal, the numbers of nodes, the numbers of features)

DESCRIPTIONS	PROMETHEUS-T	PROMETHEUS-P	WEATHERBENCH-T	WEATHERBENCH-W	NAVIER-STOKES
TRAIN SET SIZE	30000	30000	6900	6900	14000
VALIDATION SET SIZE	2000	2000	1000	1000	1000
TEST SET SIZE	2000	2000	1000	1000	1000
INPUT TENSOR	(50, 15360, 2)	(15, 32768, 3)	(12, 2048, 1)	(12, 2048, 1)	(10, 4096, 1)
OUTPUT TENSOR	(50, 15360, 2)	(15, 32768, 3)	(12, 2048, 1)	(12, 2048, 1)	(10, 4096, 1)

- **Operator learning methods:** (1). FNO (Li et al., 2021) is an efficient deep learning framework for learning solutions to partial differential equations. It operates in the frequency domain through Fourier transforms, enhancing computational efficiency. (2). F-FNO (Tran et al., 2023) is a variant of FNO that further improves model efficiency and generalization ability through factorization, suitable for a broader range of partial differential equation solutions. (3). LSM (Wu et al., 2023a) combines deep learning with mathematical operator theory for solving complex scientific computation problems, such as simulations in fluid dynamics and material science.

To adapt these models for irregular grids, we incorporate geo-FNO (Li et al., 2022b) for input/output transformation, facilitating conversion of irregular domains into regular grids.

Implementation & Evaluation Metrics. To ensure fairness, all methods train on the NVIDIA-A100 using the ADAM optimizer for MSE loss over 500 epochs, with an initial learning rate of 10^{-3}. We set the batch size to 20. We study Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM) in our research. The mathematical formulas for evaluating the indicators in decibels are shown below.

$$\mathcal{L}_{err} = \frac{1}{N} \sum_{i} \sum_{t} \left\| \mathbf{x}_i^t - \hat{\boldsymbol{x}}_i^t \right\|^2 \tag{23}$$

Here, $\mathcal{L}_{err}$ represents the mean squared error, N is the total number of samples (i.e., all combinations of i and t), $\mathbf{x}_i^t$ is the true value of the ith sample at time t, and $\hat{\boldsymbol{x}}_i^t$ is the corresponding predicted value. This formula calculates the overall mean squared error by summing and averaging the squares of the prediction errors for all samples i at all time points t.

$$\text{SSIM}(\mathbf{x}, \hat{\boldsymbol{x}}) = \frac{(2\mu_{\mathbf{x}}\mu_{\hat{\boldsymbol{x}}} + C_1)(2\sigma_{\mathbf{x}\hat{\boldsymbol{x}}} + C_2)}{(\mu_{\mathbf{x}}^2 + \mu_{\hat{\boldsymbol{x}}}^2 + C_1)(\sigma_{\mathbf{x}}^2 + \sigma_{\hat{\boldsymbol{x}}}^2 + C_2)} \tag{24}$$

In the Structural Similarity Index (SSIM) formula, $\mathbf{x}$ and $\hat{\boldsymbol{x}}$ represent the original and reconstructed (or predicted) images, respectively. The terms $\mu_{\mathbf{x}}$ and $\mu_{\hat{\boldsymbol{x}}}$ are the average luminance of the images $\mathbf{x}$ and $\hat{\boldsymbol{x}}$, while $\sigma_{\mathbf{x}}^2$ and $\sigma_{\hat{\boldsymbol{x}}}^2$ are the variances of these two images. Additionally, $\sigma_{\mathbf{x}\hat{\boldsymbol{x}}}$ denotes the covariance between the images $\mathbf{x}$ and $\hat{\boldsymbol{x}}$. To avoid division by zero, the formula includes two small constants C_1 and C_2, typically related to the dynamic range of the image data. The SSIM value ranges from -1 to 1, where 1 indicates perfect similarity between the two images. This metric is especially important in the field of image processing, particularly for image quality assessment, compression, and transmission, as it closely aligns with human visual perception of image quality. This is beneficial for visually analyzing the modeling of dynamical systems.

E. Benchmarks detials

In this section, the details of the benchmark configurations are succinctly summarized in Table 5. The data details of Prometheus are shown in Figure 6. We used a sliding window approach to generate the dataset needed for training tests.

E.1. Prometheus

FDS relies on a complex set of physical and chemical equations to model fire combustion dynamics. These equations include the Navier-Stokes equations, energy conservation equations, matter conservation equations, chemical reaction equations for combustion, radiative heat transfer models, and solid combustion and pyrolysis models. With these equations, FDS is able to

simulate fire scenarios under different conditions, including flame propagation, smoke behavior, heat release, and chemical transformations.

$$\text{Navier-Stokes Equations:} \frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \nabla \cdot \mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) + \rho \mathbf{g} \tag{25}$$

$$\text{Energy Conservation Equation:} \frac{\partial(\rho E)}{\partial t} + \nabla \cdot (\mathbf{u}(\rho E + p)) = \nabla \cdot (\kappa \nabla T) + \Phi \tag{26}$$

$$\text{Mass Conservation Equation:} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{27}$$

$$\text{Combustion Chemistry Equations:} \frac{\partial(\rho Y_i)}{\partial t} + \nabla \cdot (\rho Y_i \mathbf{u}) = \nabla \cdot (\rho D_i \nabla Y_i) + \dot{\omega}_i \tag{28}$$

$$\text{Radiative Heat Transfer Model:} \dot{q}_{\text{radiation}} = \sigma \cdot (T^4 - T^4_{\text{surroundings}}) \tag{29}$$

$$\text{Solid Combustion and Pyrolysis Models:} \frac{\partial(\rho_s T_s)}{\partial t} = \nabla \cdot (k_s \nabla T_s) + \dot{q}_{\text{pyrolysis}} + \dot{q}_{\text{combustion}} \tag{30}$$

In the equations used in the FDS, ρ represents fluid density, typically referring to the density of air or combustion gases. $\mathbf{u}$ is the fluid velocity vector, indicating the direction and speed of fluid movement in space. t stands for time. p is the fluid pressure. μ denotes the dynamic viscosity of the fluid, related to internal friction. ∇ is the vector gradient operator, used for calculating spatial changes in a field. E represents the total energy per unit mass, including both internal and kinetic energy. κ is the thermal conductivity, reflecting the material's ability to conduct heat. T signifies temperature. Φ represents other forms of energy transfer, such as energy from chemical reactions. Y_i is the mass fraction of the ith chemical component. D_i is the diffusion coefficient of that chemical component. $\dot{\omega}_i$ is the production or consumption rate of the component. σ is the Stefan-Boltzmann constant, related to thermal radiation. $T_{\text{surroundings}}$ is the surrounding environmental temperature. ρ_s and T_s represent the density and temperature of a solid, respectively. k_s is the thermal conductivity of the solid. $\dot{q}_{\text{pyrolysis}}$ and $\dot{q}_{\text{combustion}}$ represent the heat source terms for pyrolysis and combustion processes, respectively. These equations form the foundation of the FDS for simulating fire dynamics, enabling precise modeling of fluid dynamics, heat transfer, chemical reactions, and material transformations in fire scenarios. They also serve as the basis for this study.

Environmental Settings. In our study, we use the FDS to simulate tunnel scenarios. As shown in Table 6, 7 to explore fire dynamics and their impact on the environment, we focus on two main conditions: Heat Release Rate (HRR) and ventilation speed. These factors are key in influencing fire behavior and smoke movement. HRR, an important measure of fire intensity, represents the heat released per unit time. We simulate fires at different HRR levels, such as 5 MW, 10 MW, 15 MW, 20 MW, and 25 MW, to understand fire behavior from small to large scales. Ventilation speed plays a vital role in the movement and distribution of smoke in closed or semi-closed tunnel environments. We simulate fire behaviors in tunnels under various ventilation speeds, including 2 m/s, 4 m/s, 6 m/s, 8 m/s, and 10 m/s. We also randomly choose five sets of conditions, adding ventilation parameters, resulting in 30 different environmental settings for tunnel scenarios. For pool fire scenarios, we set up 25 different environments. This in-depth approach to environmental settings allows us to thoroughly assess changes in combustion dynamics under different conditions. This multi-environment simulation method is especially important for the performance of deep learning models in Out-Of-Distribution situations.

Table 6. Prometheus-T environment settings.

VELOCITY	HRR				
	5 MW	10 MW	15 MW	20 MW	25 MW
2 M/S	a_1	a_2	a_3	a_4	a_5
4 M/S	a_6	a_7	a_8	a_9	a_{10}
6 M/S	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
8 M/S	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
10 M/S	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}

Table 7. Prometheus-P environment settings.

VELOCITY	HRR				
	5 MW	10 MW	15 MW	20 MW	25 MW
2 M/S	b_1	b_2	b_3	b_4	b_5
4 M/S	b_6	b_7	b_8	b_9	b_{10}
6 M/S	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
8 M/S	b_{16}	b_{17}	b_{18}	b_{19}	b_{20}
10 M/S	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}

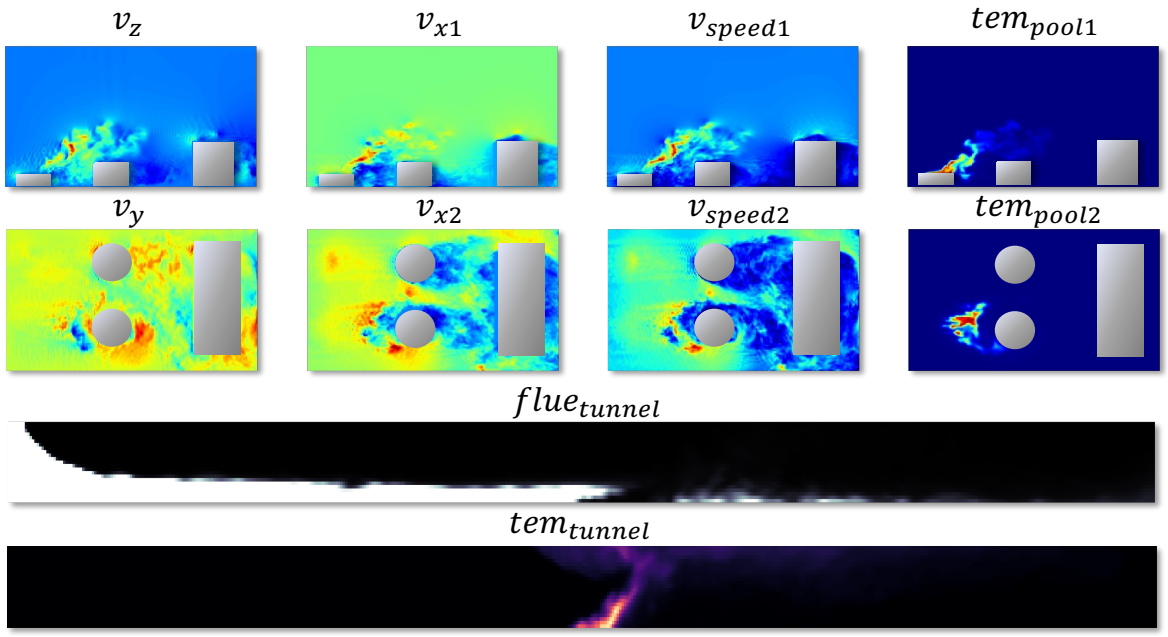


Figure 6. Visualization of simulated physical variables, the upper half is the physical field of a pool fire scene in an industrial park, and the lower half is the physical field of a tunnel fire scene.

E.2. Weatherbench

Basic Information Description. WeatherBench is a dataset for weather forecasting and climate model benchmarking. It consists of meteorological variables that are extracted from the European Center for Medium-Range Weather Forecasts (ECMWF) reanalysis data. This dataset is specifically designed for machine learning and artificial intelligence applications in weather prediction.

Environmental Settings. In our study, we analyze four key variables (Lin et al., 2022): temperature, humidity, cloud

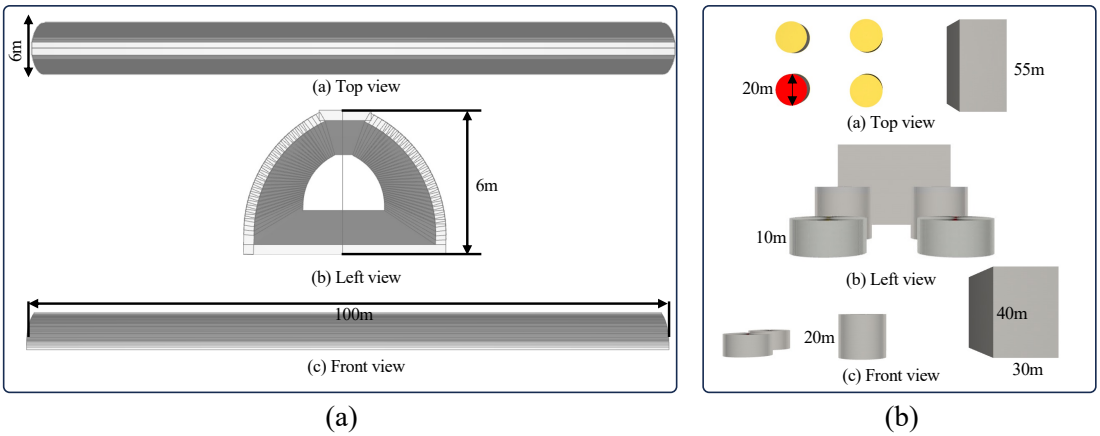


Figure 7. Tunnel and pool geometries.

cover, and surface wind components. We focus on accurately predicting temperature and surface wind components, treating the other variables as elements that constitute the predictive environment. Specifically, when predicting temperature, we construct three different environmental variable combinations: humidity and cloud cover, humidity and wind component, and cloud cover and wind component, represented as $\{c_1, c_2, c_3\}$. Similarly, in predicting the surface wind component, we consider combinations of temperature and humidity, temperature and cloud cover, and humidity and cloud cover, denoted as $\{d_1, d_2, d_3\}$. This approach allows us to better understand the interplay among the variables, thereby enhancing the accuracy of our predictions.

E.3. Navier-Stokes Equation

Basic Information Description. The dataset is generated by employing a pseudospectral method to solve the 2D Navier-Stokes equation for a viscous and incompressible flow. The equation, expressed in its vorticity form, is as follows (Li et al., 2021):

$$\begin{aligned} \partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) &= \nu \Delta w(x,t) + f(x), && x \in (0,1)^2, t \in (0,T] \\ \nabla \cdot u(x,t) &= 0, && x \in (0,1)^2, t \in [0,T] \\ w(x,0) &= w_0(x), && x \in (0,1)^2. \end{aligned} \tag{31}$$

Here, the forcing term is set to $f(x) = 0.1\left(\sin\left(2\pi\left(x_1 + x_2\right)\right) + \cos\left(2\pi\left(x_1 + x_2\right)\right)\right)$. To generate diverse solutions for training in the Banach space mapping, the initial condition w_0 is sampled from a distribution $\mu = \mathcal{N}\left(0, 7^{3/2}(-\Delta + 49I)^{-2.5}\right)$. The boundary condition applied is periodic. The viscosity coefficient ν is set at $10^{-3}, 10^{-4}, 10^{-5}$, ensuring sufficient chaos in the solution evolution over time.

Environmental Settings. The Navier-Stokes equation demonstrates its versatility in three different settings, defined by combinations of viscosity coefficients. Specifically, the combinations are $\{10^{-3}, 10^{-4}\}$, $\{10^{-3}, 10^{-5}\}$, and $\{10^{-4}, 10^{-5}\}$, each corresponding to a unique environmental setup. Thus, we represent these settings as the set $\{e_1, e_2, e_3\}$, where each element reflects a specific combination of viscosity coefficients.

F. More Showcases

Prometheus-T. The Prometheus dataset comprises two distinct scenarios, named Prometheus-T and Prometheus-P. Focusing first on Prometheus-T, we have visualized two key physical fields: temperature and smoke concentration. Additionally, we have selected the runner-up model, LSM, for comparison. This is illustrated in Figures 8. A clear observation from these visualizations is that our model outperforms LSM, particularly in terms of detail and accuracy. In contrast, the results produced by LSM appear notably smoother.

Prometheus-P. Next, we will show another set of fire simulation scenarios in benchmark Prometheus-P. We display these

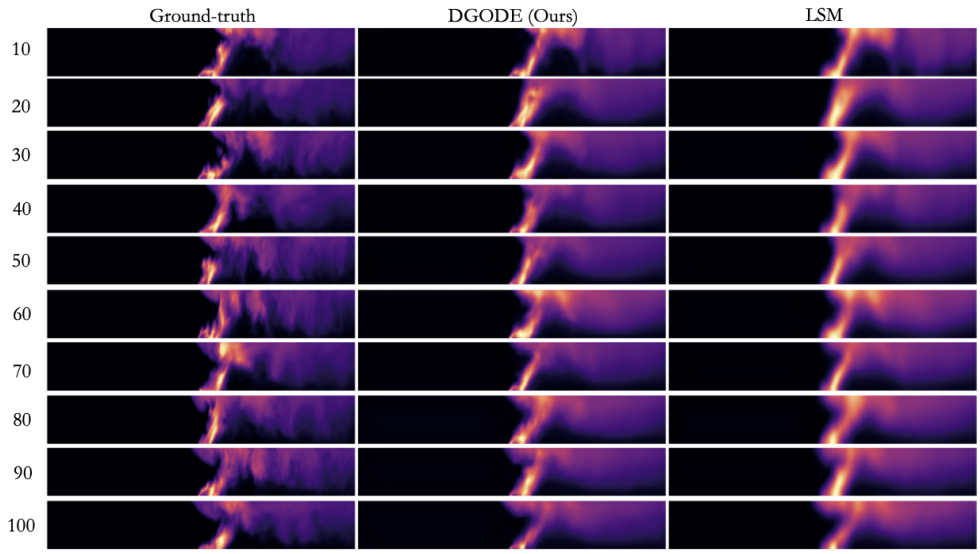


Figure 8. Visualization of temperature physical fields in Prometheus-T.

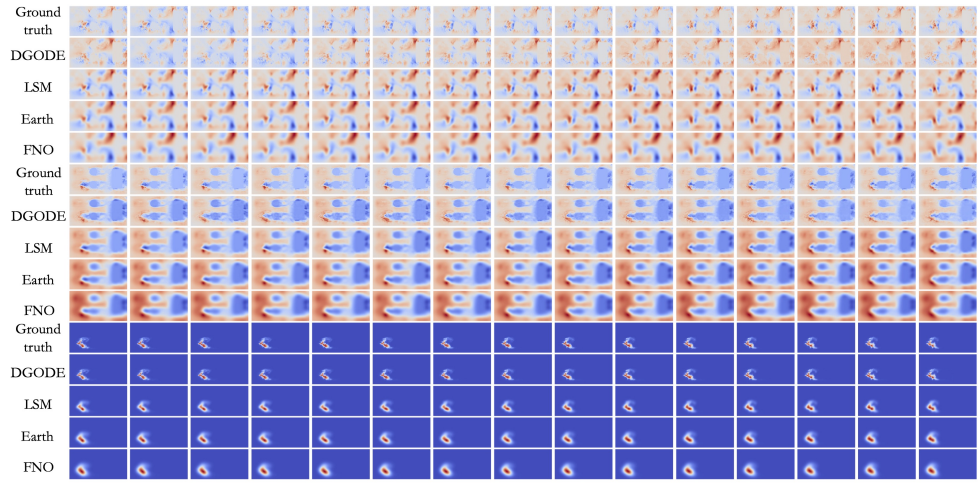


Figure 9. Overhead view visualization of the Prometheus-P.

scenarios in a top-down view Figure 9 and show the variables of speed and temperature. We can clearly see the v component and u component of speed, as well as the temperature component. Notably, the DGODE model's predictions match the real situation (Ground-truth) very closely. Compared to other models, they often show overly smooth results, but DGODE shows higher precision in handling details.

Weatherbench-T. As shown in the Figure 10, Weatherbench-T is a dataset benchmark for specifically evaluating the prediction of the temperature variable, and for a clearer visualization, we added land information, and the corresponding Error case, for highlighting the differences, and it is still noticeable that we achieved the best results with DGODE, and the visualization of the prediction is consistent with Ground truth.

Weatherbench-W. As shown in the Figure 11, Weatherbench-W is a dataset benchmark for specifically evaluating the prediction of the wind speed variable, which mainly contains two directions, x and y. In order to visualize more clearly, we have included land information, as well as the corresponding Error cases, which are used to highlight the differences, and it can still be found that we have achieved the best results for the DGODE effect, and the predicted visualization is consistent with the Ground truth.

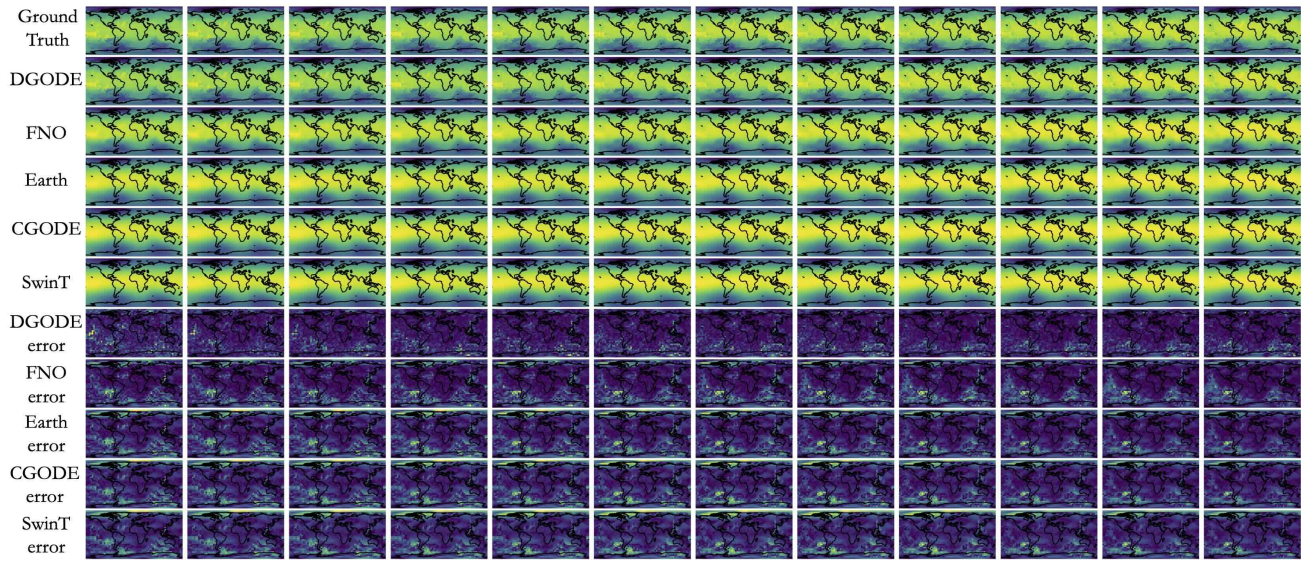


Figure 10. Visualization of temperature variables in Weatherbench-T.

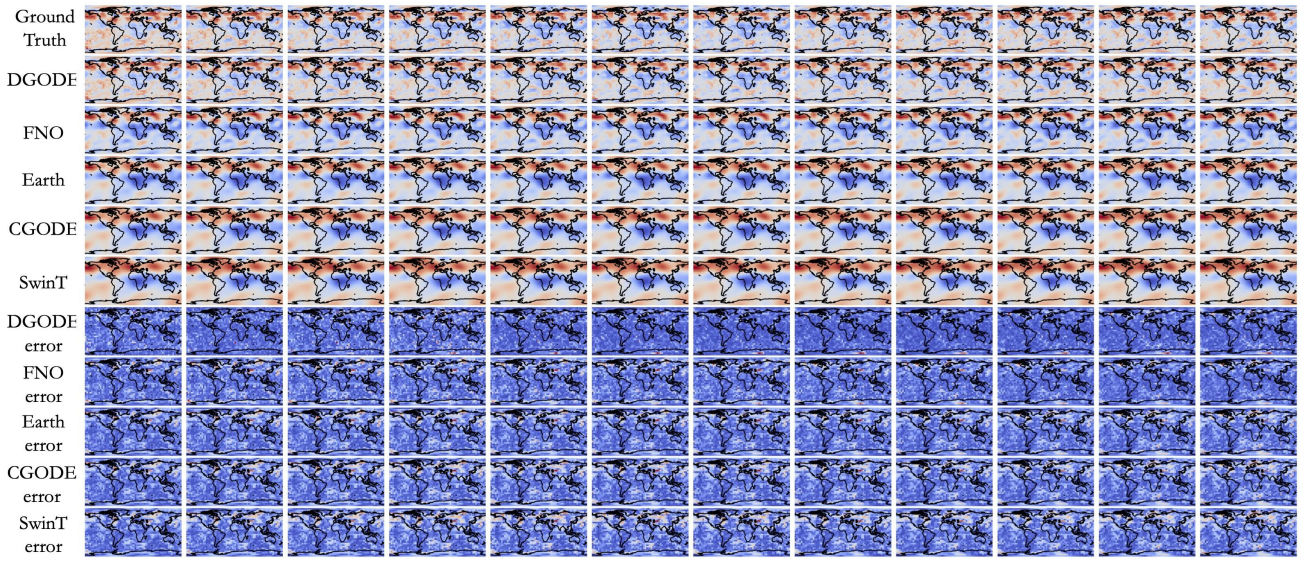


Figure 11. Visualization of wind (x) variables in Weatherbench-W.