Breaking the Discretization Barrier: Learning Continuous Physics Simulations from Partial Observations

Anonymous Authors[1]

Abstract

The modeling of complicated time-evolving physical dynamics from partial observations is a long-standing challenge. Particularly, observations can be sparsely distributed in a seemingly random or unstructured manner, making it difficult to capture highly nonlinear features in a variety of scientific and engineering problems. However, existing data-driven approaches are often constrained by fixed spatial and temporal discretization. While some researchers attempt to achieve spatio-temporal continuity by designing novel strategies, they either overly rely on traditional numerical methods or fail to truly overcome the limitations imposed by discretization. To address these, we propose `COPS`, a purely data-driven methods, to effectively model continuous physics simulation from partial observations. Specifically, we employ multiplicative filter network to fuse and encode spatial information with the corresponding observations. Then we customize geometric grids and use message-passing mechanism to map features from original spatial domain to the customized grids. Subsequently, `COPS` models continuous-time dynamics by designing multi-scale graph ODEs, while introducing a Marcov-based neural auto-correction module to assist and constrain the continuous extrapolations. Comprehensive experiments demonstrate that `COPS` advances the state-of-the-art methods in space-time continuous modeling across various scenarios.

1. Introduction

Achieving accurate global modeling and forecasting of a complex time-evolving physical system from a limited number of observations has been a long-standing challenge (Boussif et al., 2022; Yin et al., 2022). This has widespread applications in chaotic physical systems such as geophysics (Fragkiadaki et al., 2015; Song et al., 2021), atmospheric science (Defferrard et al., 2020; Stoll et al., 2020), and fluid dynamics (Zheng et al., 2018; Li et al., 2020b; Bonev et al., 2023), et al. For instance, in meteorology and oceanography, sensors are often deployed in areas with scarce or non-existent network connectivity. This renders the analysis and modeling of the system a formidable obstacle. From empirical knowledge, dynamical systems can be fully described by complicated and not yet fully elucidated partial differential equations (PDEs) (Geneva & Zabaras, 2020). Traditional numerical methods such as finite element (Gallagher et al., 1975) and finite volume (Barth et al., 2018) methods excessively rely on prior knowledge of essential PDEs and suffer from high time complexity, making it difficult to accommodate the increasing demands for grid granularity. Recently, data-driven approaches partially overcome these by integrating advanced neural networks to directly learn dynamic latent features with high nonlinearity from existing observations or simulations.

Although data-driven methods (Hao et al., 2022; Liu et al., 2023) have the capacity to learn complex relationships from observations, they still have serious limitations in changeable actual scenes. One significant challenge in modeling physical fields from partial observations is that the field may not adhere to a Cartesian grid (Pfaff et al., 2020; Ashiqur Rahman et al., 2022), and convolutional neural networks are inherently not adapted to such structures. When employing graph neural networks (Kipf & Welling, 2016), the graph topology is often determined by the locations of existing observation points, making the model difficult to generalize to unseen spatial locations. Additionally, some methods use padding or interpolation (Challu et al., 2023) to enforce spatiotemporal continuity. This may distort the inherent inductive bias of the observed data and significantly increases the computational burden in sparse data scenarios.

Recently, to better explore the space-time continuous formulation from partial observations (Iakovlev et al., 2020), a few notable works stand out. Example of continuous time and space dynamic system evolution is illustrated in Figure 1. For space-continuous learning, the multiplicative filter net-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

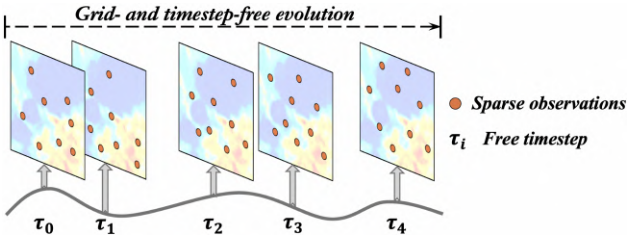

Figure 1. Example of continuous time and space dynamical system evolution. (*locations and time steps have no discrete restriction*)

work (MFN) (Fathony et al., 2020), combined with implicit neural representations (INRs) (Grattarola & Vandergheynst, 2022), has been proposed to incorporate coordinate information through linear Fourier or wavelet functions in an efficient manner. However, this approach requires iterating for hundreds of steps to obtain the corresponding implicit neural representation for new samples, severely lacking real-time applicability. Further, for time-continuous learning, neural ordinary differential equations (Neural ODEs) (Chen et al., 2018) address the limitation of fixed time extrapolation windows by learning continuous ODEs from discrete data using explicit solvers such as the Euler or Runge-Kutta methods (Mechee & H Aidi, 2022). However, if the system's nonlinearity is high, Neural ODE may struggle to capture the essential features of the complex dynamics. Additionally, Neural ODE is solved through numerical integration, which leads to errors accumulating as time progresses. As a result, conventional Neural ODEs may lead to poor stability in long-term predictions and potentially worse fitting.

To address these, we propose `COPS`, a purely data-driven methods, to effectively achieve space-time continuous prediction of dynamical systems based on partial observations. Specifically, we employ multiplicative filter networks to fuse and encode spatial information with the corresponding observations. Then we customize geometric grids and use message-passing mechanism to map features from original spatial domain to the customized grids. Subsequently, `COPS` models continuous-time dynamics by designing multi-scale graph ODEs, while introducing a local refinement feature extractor to assist and constrain the parametric evolutions. Finally, the predicted latent features are mapped back to original spatial domain through a GNN decoder.

In summary, we make the following key contributions: (i) *Encoding Mechanism.* We propose a novel encoding approach that leverages Multiplicative Filter Networks to integrate partial observations with spatial coordinate information, effectively encoding these features into a customized geometric grid. (ii) *Novel Methodology.* We introduce a multi-scale graph ODE module to model continuous-time dynamics, complemented by a neural auto-regressive correction module to assist and constrain the parametric evolution, ensuring robust and accurate predictions. (iii) *Superior Performance.* `COPS` demonstrates state-of-the-art performance on complex synthetic and real-world datasets, showcasing the possibility for accurate and efficient modeling of intricate dynamical processes and precise long-term predictions.

2. Related Work

2.1. Deep Learning for Physical Simulations

Recent advancements in deep neural networks have established them as effective tools for addressing the complexities of dynamics modeling (Gao et al., 2022; Yin et al., 2022), demonstrating their ability to efficiently capture the intricacies of high-dimensional systems. Physics-Informed Neural Networks (PINNs) (Kharazmi et al., 2019; Krishnapriyan et al., 2021), which optimize weights by minimizing the residual loss derived from the PDE, have received considerable attention due to their flexibility in tackling a wide range of data-driven solutions (Li et al., 2020a). Recently, neural operators (Kovachki et al., 2023; Li et al., 2020b; Wu et al., 2024b; Raonic et al., 2024) map between infinite–dimensional function spaces by replacing standard convolution with continuous alternatives. Specifically, they utilize kernels in fourier space (FNO) (Li et al., 2020b) or graph structure (GNO) (Li et al., 2020c) to learn the correspondence from the initial condition to the solution at a fixed horizon. However, most existing methods are limited by a static observation grid, restricting their ability to evaluate points outside the training domain and confining queries to fixed time intervals. In this work, we aim to overcome discretization and achieve spatio-temporal modeling in a continuous way.

2.2. Advanced Space-Continuous Modeling

Interpolation has been widely adopted in numerous applications (Challu et al., 2023; Iakovlev et al., 2024), which also holds wide prospects in spatio-temporal modeling. For instance, researchers employed interpolation as a post-processing technique to generate continuous predictions. Recently, MAgNet (Boussif et al., 2022) proposes to predict the dynamical solution after interpolating the observation graph in latent space to new query points. Another approach for space-continuous or grid-independence modeling is a kind of coordinate-based neural networks, called Implicit Neural Representations (INRs) (Grattarola & Vandergheynst, 2022). Typically, INRs take the spatial coordinates as inputs along with other conditions, which can be utilized to enhance real space-continuous modeling and dynamical evolution predicting. For instance, DINO (Yin et al., 2022) overcomes the limitation of failing to generalize to new grids or resolution via a time-continuous dynamics model of the underlying flow. MMGN (Luo et al., 2024) learns relevant basis functions from sparse observations to

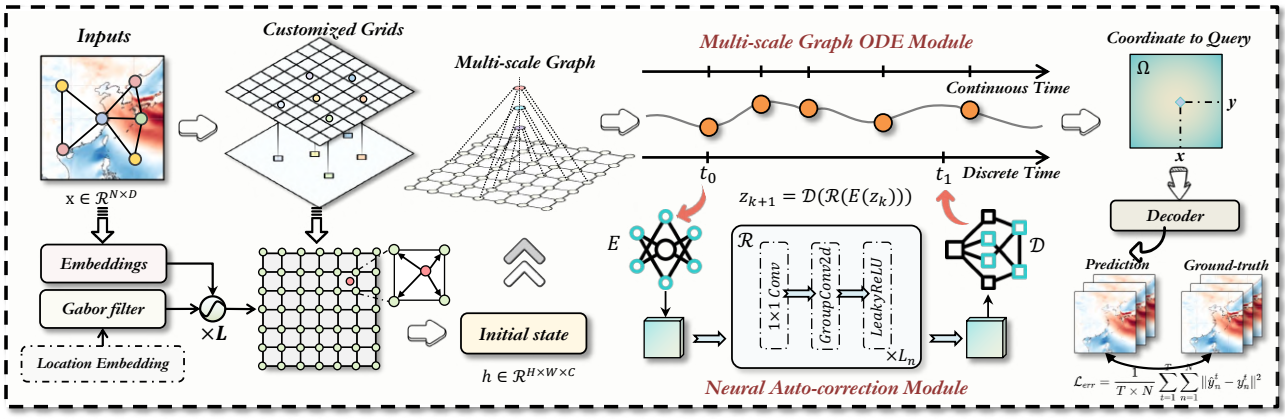

Figure 2. **The overview of COPS.** *Stage 1:* Employ multiplicative filter network to encode the initial representation, and map it to the customized grids with message passing scheme. *Stage 2:* Model the latent dynamics with multi-scale Graph ODE module and auto-correction module in a time-continuous way. *Stage 3:* Extrapolate results for arbitrary future time step and coordinates.

obtain continuous representations after actorizing spatio-temporal variability into spatial and temporal components.

3. Methodology

3.1. Problem Definition

In this work, we focus on modeling spatio-temporal dynamical systems through partial observations. The systems are defined over the temporal domain $\mathcal{T}$ and spatial domain Ω. The observations are recorded at L arbitrary consecutive time points $t_{1:L} := (t_1, \ldots, t_L) \in \mathcal{T}$, and N arbitrary local sensor measurements at locations $x_i \in \Omega, i = (1, \ldots, N)$. To adapt to general real-world scenarios, we assume that all dynamics are learned from the initial observation $u(t_0) = (u(x_1, t_0), \ldots, u(x_N, t_0))^T$. Moreover, the values of time steps and observation locations may vary across different observed trajectories during the inference phase. Thus, for prediction, our goal is to learn a neural function $\mathcal{Q}(\cdot)$, which maps the initial observation at the first time step to future dynamical predictions at arbitrary time steps and locations:

$$\mathcal{Q}(u(t_0, x_0); \mathcal{T}, \Omega) \to \mathcal{Y}(t_i, x_j),\ t_i \in \mathcal{T},\ x_j \in \Omega. \quad (1)$$

where t_i and x_j represent arbitrary future time step and location, respectively.

3.2. Framework Overview

In this section, we introduce the proposed COPS, which achieves the continuous spatio-temporal modeling from partial observations within complicated dynamical systems. We aim to simultaneously overcome the discretization limitations in both the spatial and temporal domains. To better align with the form of numerical solutions, we begin with partial observations of the initial state and apply constraints based on real observations during the continuous evolution process. An overview of our COPS can see in Figure 2.

3.3. Initial Encoding via Multiplicative Filter Network

Initial Encoder. Further, we follow the multiplicative filter network and choose nonlinear gabor filter $g_\kappa(\cdot)$ (Fathony et al., 2020) to generate a continuous global frequency representation. The formulation is as follows:

$$g_\kappa(x) = \exp\left(-\frac{\gamma^{(\kappa)}}{2}\left\|x - \mu^{(\kappa)}\right\|_2^2\right) \sin\left(W_g^{(\kappa)} x + b_g^{(\kappa)}\right), \quad (2)$$

where $\mu^{(\kappa)} \in \mathbb{R}^{d_h}$ and $\gamma^{(\kappa)} \in \mathbb{R}^{d_h \times d_x}$ denote the respective mean and scale term of the κ-th Gabor filter. Subsequently, the filters are multiplied by the linear transformation of the previous layer's embedding and the historical sequence representation. The iteration process is defined as follows:

$$\rho_i^1 = g_1(x_i), \quad \mathcal{H}_\phi(u_i, x_i) = u_i + W_i^M \rho_i^M + b_i^M,$$
$$\rho_i^{(\kappa+1)} = (u_i + W_i^\kappa \rho_i^\kappa + b_i^\kappa) \odot g_\kappa(x), \kappa = 1, \ldots, M-1, \quad (3)$$

where $\odot$ denotes element-wise multiplication, W_i^κ and b_i^κ are the learnable parameters. Finally, $\mathcal{H}_\phi(u_i, x_i)$ is the obtained vectors that contain complicated coordinate information and feature of node i.

Customized Grids and Message Passing Scheme. For our observational data, the spatial domain is non-discrete. Recent works like MeshGraphNet (Pfaff et al., 2020) rely on fixed meshes, which presents challenges when generalizing to previously unseen coordinate points. we propose a novel customized grid mapping approach. Specifically, for any arbitrary coordinate, it can be mapped onto a uniform grid structure. Let $p_i = (x_i, y_i)$ denotes an arbitrary point in the continuous $2D$ space, we will discover the appropriate grid cell according to it. To enrich the representation, we connect each point p_i to the four vertices $\{v_{i1}, v_{i1}, v_{i3}, v_{i4}\}$

of the associated grid cell.

Based on the encoded initial state $h_i^0 = \mathcal{H}_\phi(\boldsymbol{u}_i, \boldsymbol{x}_i)$ and the customized grids, the next step involves a message-passing scheme that propagates the features from the original continuous space to the customized grids. To further preserve spatial continuity, the scheme encodes and incorporates the relative positions of the connected points during message-passing. The transformation is as follows:

$$h_i^{k+1} = h_i^k + \sum_{j \in \mathcal{N}(p_i)} W_{ij}^k \left(h_j^k - h_i^k + \phi(p_i, p_j) \right) + b^k, \quad (4)$$

where W_{ij}^k and b^k are learnable parameters, and $\phi(\cdot)$ denotes the relative position coding. In this way, we collaboratively encode both the feature information and rich spatial information onto the customized grids, completing the initial encoding process.

3.4. Latent Dynamics Modeling

Further, we view high-dimensional features on the customized grids as latent states and aim to efficiently model complicated dynamics in a time-continuous way.

Multi-scale Graph ODE. To capture spatio-temporal dynamics over complex domains, we build a hierarchical graph representation inspired by GraphCast (Lam et al., 2023) on top of the customized grids. Our objective is to progressively encode both coarse global behaviors and refined local structures. Concretely, let $\{\mathcal{G}^{(s)}\}_{s=1}^S$ be a collection of graphs, where $\mathcal{G}^{(S)}$ coincides with the finest-scale grid obtained from the initial encoding process. Specifically, within each graph $\mathcal{G}^{(s)}$, we connect each node to its spatial neighbors by "jump adjacency" on the underlying 2D grid. We then coarsen these into multiple scales, so that $\mathcal{G}^{(S-1)}, \ldots, \mathcal{G}^{(1)}$ provide successively simpler graph topologies.

To perform hierarchical message passing, we process each scale independently and then fuse the features using an attention mechanism. For each scale s, the message passing is defined as follows:

$$\begin{aligned} h_i^{'(l,s)} &= \text{AGGREGATE}\left(\left\{ h_j^{(l-1,s)} : j \in \mathcal{N}^{(s)}(i) \right\} \right) \\ h_i^{(l,s)} &= \text{COMBINE}\left(h_i^{(l-1,s)}, h_i^{'(l,s)} \right), \end{aligned} \quad (5)$$

where $\mathcal{N}^{(s)}(i)$ denotes the neighbors of node s at scale s. After propagating messages within each scale, we employ an attention mechanism to fuse the features across scales.

$$h_i^{(l)} = \sum_{s=1}^{S} \alpha_i^{(s)} h_i^{(l,s)}, \ \alpha_i^{(s)} = \text{Attention}\left(h_i^{(l,s)}, h_i^{(l-1)} \right), \quad (6)$$

where $\alpha_i^{(s)}$ denotes the attention weight. Building on this hierarchical representation, we parametrize a coupled graph ODE to model the continuous-time evolution of it's latent state.

$$\frac{dh_i^t}{dt} = \Phi\left(h_1^t, h_2^t, \cdots, h_N^t, \mathrm{G}, \Theta \right). \quad (7)$$

Here, Φ denotes the multi-scale message passing function, G represents the hierarchical graph structure, and Θ encapsulates the model parameters. Given the ODE function Φ, the node initial values, and the corresponding contextual vector, the latent dynamics can be solved by any ODE solver like Runge-Kutta (Schober et al., 2014).

$$h_i^{t_1} \cdots h_i^{t_T} = \text{ODESolve}(\Phi, [h_1^0, h_2^0 \cdots h_N^0]). \quad (8)$$

This formulation allows the model to theoretically obtain the hidden state at any arbitrary timestep, provided that the Neural ODE fits the underlying dynamics well.

Neural Auto-correction. Although the multi-scale graph ODE framework provides a global continuous-time evolution, it still relies on the strong assumption that a learnable ODE exists to model the concrete physical dynamics. Moreover, neural ODEs relying on numerical solvers may fail to capture the system's intrinsic nonlinear features, making it difficult to handle error divergence during evolution. To address this, we introduce a neural auto-correction module that imposes a discrete dynamical regime in the latent space. Thus we can capture complex corrections at selected time steps in a heuristic Markov chain manner.

At each correction step, a convolutional encoder first compresses the current latent representation into a compact state. Subsequently, after the discrete evolution block, a transposed convolution-based decoder reconstructs the refined latent field. This operation acts as a "Markov state observer" (Choromanski et al., 2020), learning a discrete single-step mapping in a more compact latent space. The formulation is as follows:

$$E(\cdot) = \sigma(\text{LN}(\text{Conv2d}(\cdot))), \ D(\cdot) = \text{Tanh}(\text{UnConv2d}(\cdot)). \quad (9)$$

Further, between the encoder and decoder lies a neural block that embodies a discrete transition operator similar to a Markov kernel. Technically, it consists of a 1×1 Conv2D, which strips away extraneous channels to focus on the most critical latent features. Then, it is followed by parallel GroupConv2D (Tan et al., 2023) operators to process these features in multiple subspaces. We utilize $\mathcal{R}$ to represent the combination of the 1×1 Conv2D and the parallel GroupConv2Ds. Then the whole evolution can be viewed as:

$$z(k+1) \ = \ D(\mathcal{R}\big(E(z(k))\big)), \quad (10)$$

where $z(k)$ denotes the latent embedding at discrete step k. Crucially, this transformation depends only on the current embedding $z(k)$, thus fulfilling a Markov property at

each correction step. By encapsulating key spatiotemporal features in $z(k)$ and evolving them into $z(k+1)$ via $\mathcal{R}(\cdot)$, the neural auto-correction module adaptively regularizes the global continuous-time solution. On one hand, it provides a local perspective on how errors can be contained and corrected in each interval. On the other hand, it retains a sufficient memory of the system's evolving states without requiring full historical trajectories.

Inference Strategy. During inference, we employ an iterative strategy that combines the Multi-scale Graph ODE Module and the Neural Auto-correction Module to generate continuous accurate predictions. Starting from the initial state $h_i(t_0)$, the model proceeds as follows:

$\star$ *Continuous Evolution*: The multi-scale Graph ODE module evolves the latent states continuously from t_k to t_{k+1}:

$$h_i(t_{k+1}) = h_i(t_k) + \int_{t_k}^{t_{k+1}} f_\theta\left(h_i(t), \{h_j(t)\}_{j\in N(i)}, t\right) dt. \tag{11}$$

$\star$ *Discrete Correction*: At each discrete time step t_{k+1}, the Neural Auto-correction Module refines the predicted state:

$$h_i^\omega(t_{k+1}) = h_i(t_{k+1}) + \lambda r_\psi\left(h_i(t_k)\right), \tag{12}$$

where r_ψ denotes the neural auto-correction module, and λ is the hyperparameter for balance. The corrected state $h_i^\omega(t_{k+1})$ is used as the initial state for the next time step, and the process repeats until the final prediction time is reached. This inference strategy ensures that the model can generate accurate and stable predictions over long time horizons, while maintaining the continuity and smoothness of the learned dynamics.

3.5. Decoder and Optimization

Decoder. Upon obtaining the time-continuous latent states on the customized grids, we project them back to the original continuous spatial domain for final prediction. Specifically, for a query coordinate $q_m = \{x_m, y_m\}$, we first identify the corresponding grid cell v_m and establish connections to its four vertices, denoted $\{v_{m1}, v_{m2}, v_{m3}, v_{m4}\}$. We then initialize the representation of q_m with a single step Gabor filter transformation $h_m = g_1(q_m)$, and then perform a message-passing update just as in Eq 4. Further, we can obtain the corresponding predictions through a multi-layer perception (MLP) decoder $f_{dec}(\cdot)$.

Optimization. For each training sample, we split it into two components along the timeline, and the former is utilized to learn historical dynamics and predict future dynamics. To optimize the whole framework, we minimize the training error compared with the ground truth as follows:

$$\mathcal{L}_{err} = \frac{1}{T \times N} \sum_{t=1}^{T} \sum_{n=1}^{N} \|\widehat{y}_n^t - y_n^t\|^2. \tag{13}$$

3.6. Theoretical Analysis

Theorem 3.1. ***Error Bounding via Autoregressive Corrections.*** *Consider a dynamical system governed by the Neural ODE model:*

$$\frac{d}{dt}y(t) = \mathcal{F}(y(t), \theta), \tag{14}$$

with initial condition $y(0) = x(0)$, where $\mathcal{F}$ is a smooth, L-Lipschitz continuous function. Let $\mathcal{P}(t)$ represent the true system trajectory and $\mathcal{F}(t)$ denote the predicted trajectory. We define the error between the true and predicted states as:

$$e(t_k) = \|\mathcal{P}(t_k) - \mathcal{F}(t_k)\|. \tag{15}$$

At each correction step t_k, the error is mitigated by applying a correction operator $\mathcal{R}(y(t_k^-))$ such that:

$$y(t_k^+) = y(t_k^-) + \mathcal{R}(y(t_k^-)), \tag{16}$$

with $\|\mathcal{R}\| \leq \gamma$, where γ is a constant bounding the magnitude of the correction. After applying corrections, the error is reduced significantly at each time step, leading to the following error bound:

$$\|e(t)\| \;\leq\; C \cdot \alpha^{\lfloor t/\Delta t_{\text{corr}} \rfloor}, \tag{17}$$

where $C > 0$ is a constant depending on the initial error and the system's characteristics, and $\alpha \in (0, 1)$ is a constant that quantifies the error decay per correction step.

4. Experiment

4.1. Experimental Settings

Datasets. To illustrate the property and efficacy of the extrapolations obtained from COPS, we conduct experiments on diverse synthetic and real-world datasets. For the synthetic datasets, we first choose **Navier-Stokes** (Li et al., 2020b) and **Rayleigh–Bénard Convection** (Wang et al., 2019), which are directly generated by numeric PDE solvers. Then, we select **Prometheus** (Wu et al., 2024a), which is a large-scale combustion dataset simulated with industrial software. For real-world datasets, we choose **WeatherBench** (Rasp et al., 2023), a dataset for weather forecasting and climate modeling. We also select **Kuroshio** (Wu et al., 2024b), which provides vector data of sea surface stream velocity from the Copernicus Marine Service. See Appendix B for detailed descriptions of all these datasets.

Baselines. We evaluate our model against three baselines representing the state-of-the-art in continuous modeling. **MAgNet** (Boussif et al., 2022) employs an "Encode-Interpolate-Forecast" scheme. Specifically, MAgNet employs the nearest neighbor interpolation technique to generalize to new query points. Subsequently, it forecasts the evolutionary trends by leveraging a GNN-based message

Table 1. To evaluate the spatial inquiry power of our method, we vary the number of available measurement points in the data for training from 25%, 50% and 75% amount of observations. We report MSE compared to the ground truth solution.

Model		Navier-Stokes			Kuroshio			Prometheus			Weatherbench		
		s=25%	s=50%	s=75%	s=25%	s=50%	s=75%	s=25%	s=50%	s=75%	s=25%	s=50%	s=75%
MAgNet	In-s	3.164E-02	2.775E-02	2.268E-02	7.104E-03	6.523E-03	4.845E-03	1.694E-02	1.273E-02	1.186E-02	3.635E-02	3.028E-02	2.483E-02
	Ext-s	5.846E-02	4.285E-02	3.114E-02	9.464E-03	8.173E-03	7.518E-03	2.775E-02	2.322E-02	1.724E-02	5.356E-02	4.972E-02	3.295E-02
DINo	In-s	1.074E-02	9.564E-03	7.554E-03	3.737E-03	3.332E-03	2.884E-03	1.223E-02	8.005E-03	5.657E-03	2.365E-02	2.186E-02	1.922E-02
	Ext-s	2.537E-02	1.764E-02	9.248E-03	5.923E-03	5.271E-03	4.487E-03	1.784E-02	1.246E-02	8.324E-03	3.823E-02	2.746E-02	2.588E-02
ContiPDE	In-s	5.456E-03	4.176E-03	3.825E-03	2.352E-03	1.947E-03	1.503E-03	8.462E-03	6.084E-03	4.763E-03	1.542E-02	1.294E-02	1.056E-02
	Ext-s	9.523E-03	7.975E-03	6.231E-03	3.763E-03	2.531E-03	2.084E-03	1.125E-02	8.355E-03	6.674E-03	2.172E-02	1.653E-02	1.474E-02
Ours	In-s	**2.964E-03**	**2.253E-03**	**1.464E-03**	**1.285E-03**	**7.253E-04**	**5.253E-04**	**5.176E-03**	**3.722E-03**	**2.746E-03**	**8.766E-03**	**5.249E-03**	**3.585E-03**
	Ext-s	**5.743E-03**	**4.571E-03**	**2.872E-03**	**2.381E-03**	**1.577E-03**	**9.272E-04**	**8.264E-03**	**5.142E-03**	**4.287E-03**	**1.769E-02**	**1.056E-02**	**8.472E-03**

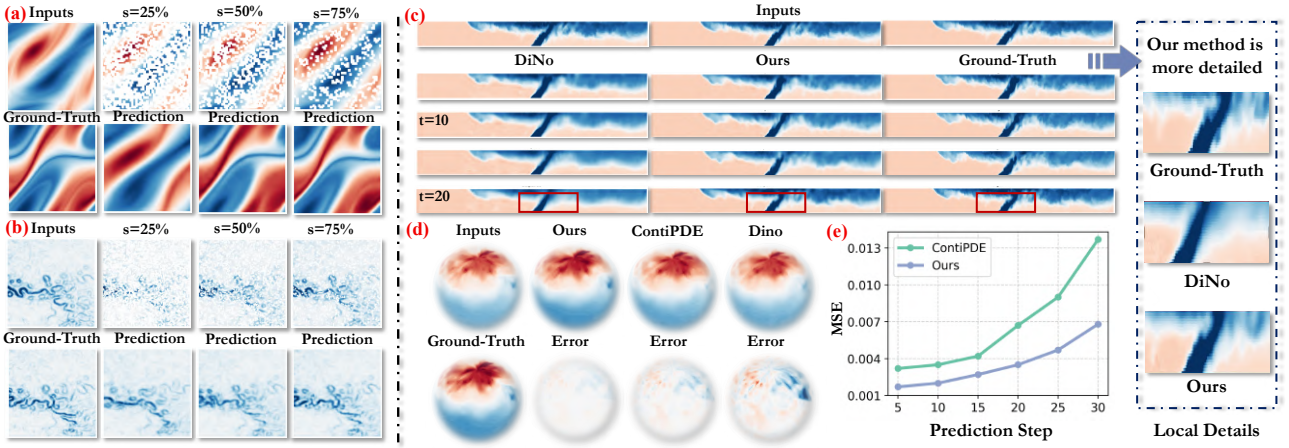

Figure 3. (a) Figure (a) and (b) shows the perdiction performance based on observations with **diverse ratio of subsampling** (25%, 50%, 75%) on the Navier-Stokes and Kuroshio dataset. Figure (c) demonstrates **long-term extrapolation beyond the training horizon** on the Prometheus dataset, and figure (e) shows the evolution of prediction errors over time steps, revealing the increasing error with longer prediction steps. Figure (d) illustrates **continuous-time prediction for intermediate points between discrete time steps** on the WeatherBench dataset, showcasing the model's ability to predict in non-training time scales.

passing neural network. **DINO** (Yin et al., 2022) learns PDE's flow to forecast its dynamical evolution by leveraging a spatial implicit neural representation modulated by a context vector and modeling continuous-time evolution with a learned latent ODE. This is the closest approach to our method. **ContiPDE** (Steeven et al., 2024) formulates the task as a double observation problem. It utilizes recurrent GNNs to roll out predictions of anchor states from the IC, and employs spatio-temporal attention observer to estimate the state at the query position from these anchor states.

Tasks. We assess the performance of COPS across diverse forecasting tasks to evaluate their efficacy in various scenarios. We use the mean squared error (MSE) as the performance measurement. (i) **Sparse Flexibility** - Gauging the effectiveness of predicting global field evolution based on observations with diverse degrees of sparsity (subsampling ratio of 25%, 50%, 75%). (ii) **Time Flexibility** - Evaluating our model's performance in extrapolating beyond the training horizon. Here we design two experimental setups: long-term extrapolation beyond the training horizon, and continuous-time prediction for intermediate points between discrete time steps. (iii) **Resolution Generalization** - Investigating the performance of generalizing to new resolution (up or down). (iv) **Super-resolution** - Investigating the model's effectiveness of super-resolution query and extrapolation. (v) **Noise Robustness** - Exploring the robustness of our model against varying noise ratios. (vi) **Ablation Study** - Investigating contributes of each key component to the performance of COPS.

Implementation. For all datasets, we normalize the data and split it into training, validation, and test sets with a ratio of 7:2:1. To ensure fairness, we use partial observations of the initial state as input and supervise the training with future real observations at discrete time steps (only using observed future values). In evaluating spatial continuity, we perform subsampling of the full initial state at rates of {25%, 50%, and 75%}, assessing both observed and unobserved points. For temporal continuity, we evaluate three criteria based on different subsampling rates: predictions within the training horizon, long-term predictions beyond the training

Table 2. We evaluate the temporal extrapolation performance of our method. Models are trained and evaluated with observation subsampling ratio of 50% and 100%. We employ three kinds of extrapolation, containing: (1) extrapolation within discrete training horizon (In-t); (2) extrapolation exceeding discrete training horizon (Ext-t); and (3) extrapolation of intermediate points between time steps (Con-t). We report MSE compared to the ground truth solution.

	Navier-Stokes			Rayleigh–Bénard Convection			Prometheus		
	In-t	Ext-t	Con-t	In-t	Ext-t	Con-t	In-t	Ext-t	Con-t
				$s = 50\%$ *subsampling ratio*					
MAgNet	1.253E-02	2.601E-02	–	9.421E-03	1.335E-02	–	1.138E-02	1.824E-02	–
DINo	7.651E-03	1.074E-02	9.971E-03	5.748E-03	8.472E-03	7.342E-03	8.321E-03	1.117E-02	8.852E-03
ContiPDE	5.054E-03	8.342E-03	6.447E-03	3.744E-03	3.814E-03	2.908E-03	5.435E-03	8.154E-03	6.637E-03
Ours	**2.832E-03**	**5.764E-03**	**4.135E-03**	**1.526E-03**	**2.328E-03**	**1.765E-03**	**3.374E-03**	**6.678E-03**	**5.355E-03**
				$s = 100\%$ *subsampling ratio*					
MAgNet	7.429E-03	1.273E-02	–	4.837E-03	7.235E-03	–	7.938E-03	1.024E-02	–
DINo	4.352E-03	7.374E-03	5.271E-03	2.248E-03	4.872E-03	3.742E-03	5.721E-03	8.217E-03	6.288E-03
ContiPDE	2.655E-03	4.742E-03	3.847E-03	1.244E-03	3.214E-03	2.308E-03	3.835E-03	6.554E-03	4.037E-03
Ours	**1.132E-03**	**2.364E-03**	**1.735E-03**	**6.522E-04**	**1.327E-03**	**1.061E-03**	**2.877E-03**	**5.174E-03**	**3.852E-03**

horizon, and continuous-time predictions for intermediate points between discrete time steps. More detailed implementations are illustrated in Appendix C.

4.2. Main Results

Space Flexibility. To evaluate the spatial querying ability of our method, we adjust the number of available measurement points by using different proportions of observation points (25%, 50%, 75%) in the training data. We report the MSE compared to the ground truth on four datasets, as shown in Table 1. From the table, we see that our method (Ours) significantly outperforms the baseline methods on all datasets and at all observation sparsity levels. On the Navier-Stokes dataset, with only 25% of the observation data, our method achieves MSEs of 2.964E-03 and 5.743E-03 under the In-s and Ext-s settings, which are much better than ContiPDE (5.456E-03 and 9.523E-03) and DINo (1.074E-02 and 2.537E-02). As the observation ratio increases to 50% and 75%, our model's performance further improves. The MSEs for In-s decrease to 2.253E-03 and 1.464E-03, and for Ext-s decrease to 4.571E-03 and 2.872E-03. On the Kuroshio dataset, even with only 25% observation data, our method achieves MSEs of 1.285E-03 and 2.381E-03 under In-s and Ext-s, significantly better than other methods. For example, ContiPDE's MSEs are 2.352E-03 and 3.763E-03, and DINo's are 3.737E-03 and 5.923E-03. As the observation ratio increases, our method's MSE further decreases, and its accuracy becomes higher. On the Prometheus and WeatherBench datasets, our method also performs excellently. Especially on the WeatherBench dataset, when the observation ratio is 75%, our method achieves MSEs of 3.585E-03 and 8.472E-03 under In-s and Ext-s, much lower than other baseline methods. These results show that our method effectively recovers the global scene from partial observations on different datasets and observation sparsity levels. It demonstrates robustness in handling sparse data and spatial generalization.

Time Flexibility. To evaluate our method's performance in temporal extrapolation, we train and evaluate the model with observation subsampling ratio of 50% and 100%. We use three extrapolation modes: (1) Extrapolation within the discrete training range (In-t); (2) Extrapolation beyond the discrete training range (Ext-t); (3) Extrapolation at intermediate points between time steps (Con-t). We report the mean squared error (MSE) on the Navier-Stokes, Rayleigh–Bénard Convection, and Prometheus datasets, as shown in Table 2. From Table 2, we see that our method achieves the best performance on all datasets and extrapolation settings. For example, on the Navier-Stokes dataset with a subsampling ratio of 50%, our method achieves MSEs of 2.832E-03 (In-t), 5.764E-03 (Ext-t), and 4.135E-03 (Con-t). These are significantly better than ContiPDE (5.054E-03, 8.342E-03, 6.447E-03) and DINo (7.651E-03, 1.074E-02, 9.971E-03). When the subsampling ratio increases to 100%, the performance improves further. Our method's MSEs decrease to 1.132E-03 (In-t), 2.364E-03 (Ext-t), and 1.735E-03 (Con-t). On the Rayleigh–Bénard Convection dataset, our method also achieves MSEs of 6.522E-04 (In-t) and 1.327E-03 (Ext-t), significantly better than other baseline methods. This shows that our method can make accurate predictions within the discrete training range. It can also effectively extrapolate to time points beyond the training range. When we perform continuous-time prediction at intermediate points between time steps (Con-t), our method also performs excellently. It achieves the smallest prediction error. Overall, these results demonstrate the strong capabil-

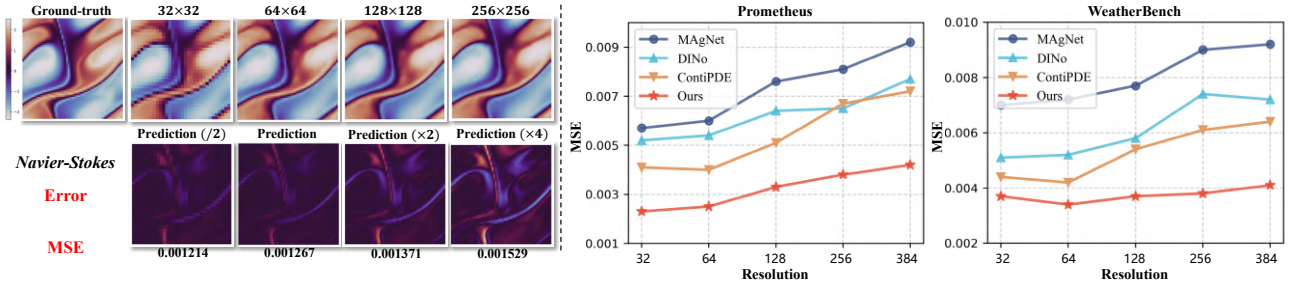

Figure 4. Left shows the inference performance for super-resolution on Navier-Stokes dataset. *Right* demonstrates the resolution generalization capability compared with MAgNet, DINo, and ContiPDE.

ity of our method in continuous-time modeling. It provides accurate and reliable predictions under different temporal extrapolation settings.

Visualization and Analysis. Figure 3 provides several clear visualization cases, highlighting COPS's effectiveness in various challenging tasks. Specifically, in figure 3 (a) and (b), we find that our COPS can effectively model the entire physical field and learn the evolution trends of unobserved points only with partial observations. When the subsampling ratio reaches 75%, our model can achieve a precise prediction on Navier-Stokes and Kuroshio datasets. As observed in figure 3 (c) and (e), we find that when compared to DINo, our COPS demonstrates better long-term prediction performance on Prometheus dataset. Additionally, in figure 3 (d), our COPS shows the optimal performance when predicting the results between discrete time steps.

4.3. Model Analysis

Analysis of Super-resolution. In the super-resolution experiment presented in Figure 4 (*left*), the results illustrate the inference performance on the Navier-Stokes dataset across resolutions of 32×32, 64×64, 128×128, and 256×256. The results consistently demonstrate improved prediction quality as resolution increases. Compared to baseline models (MAgNet, DINo, and ContiPDE), the proposed model effectively reduces prediction error at high resolutions, maintaining a lower MSE and providing clearer, more accurate predictions of the original Navier-Stokes patterns.

Resolution Generalization. As observed in Figure 4 (*right*), the folding diagrams depict the resolution generalization capabilities for both Prometheus and WeatherBench datasets. The proposed COPS consistently outperforms the comparative models (MAgNet, DINo, ContiPDE) at all resolution levels, achieving the lowest MSE value. This result highlights COPS's robustness and effectiveness in handling resolution variations. The findings emphasize the model's ability to generalize across different resolutions, which is crucial for applications requiring detailed and precise predictions in complex dynamic systems.

Table 3. Ablation study on Kuroshio. (Report MSE)

	In-s / In-t	Ext-s / In-t	In-s / Ext-t	Ext-s / Ext-t
w/o MFN	1.027E-03	1.946E-03	2.458E-03	3.149E-03
w/o MGO	1.594E-03	2.357E-03	3.053E-03	4.175E-03
w/o NAC	1.428E-03	2.216E-03	2.742E-03	3.657E-03
Ours	**7.253E-04**	**1.577E-03**	**2.161E-03**	**2.938E-03**

Ablation Study. To evaluate the contribution and importance of each component in the proposed COPS, we conduct corresponding ablation experiments, and use MSE error as metric. Our model variants are as follows: (1) **COPS w/o MFN**, we remove the multiplicative filter network and use an MLP. (2) **COPS w/o MGO**, we remove the multi-scale Graph ODE module. (3) **COPS w/o NAC**, we remove the neural auto-correction module. Table 3 shows the results of our ablation study. The results of the ablation experiments show that removing any component results in a decrease in predictive performance, further proving the critical role of these components in COPS. Specifically, the performance decline after removing the multi-scale Graph ODE module is particularly evident, proving its importance in capturing dynamic representations.

5. Conclusion

In this paper, we introduce COPS, a novel data-driven framework for modeling continuous spatio-temporal dynamics from partial observations. To overcome discretization, we first customize geometric grids and employ multiplicative filter network to fuse and encode spatial information with the corresponding observations. After encoding to the concrete grids, COPS model continuous-time dynamics by designing multi-scale graph ODEs, while introducing a Marcov-based neural auto-correction module to assist and constrain the continuous extrapolations. Extensive experiments on both synthetic and real-world datasets demonstrate the superior performance of COPS, which underlines the potential to provide a robust framework for accurate long-term predictions in the face of sparse and unstructured data.

Impact Statement

This work aims to provide a novel framework for continuous spatio-temporal modeling from partial observations. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here. Our method, COPS, alleviates the heavy reliance on space-time discretization, thus opening avenues for more efficient and flexible simulations in domains such as fluid mechanics, climate modeling, and geophysics.

References

Ashiqur Rahman, M., Ross, Z. E., and Azizzadenesheli, K. U-no: U-shaped neural operators. *arXiv e-prints*, pp. arXiv–2204, 2022.

Barth, T., Herbin, R., and Ohlberger, M. Finite volume methods: foundation and analysis. *Encyclopedia of computational mechanics second edition*, pp. 1–60, 2018.

Bonev, B., Kurth, T., Hundt, C., Pathak, J., Baust, M., Kashinath, K., and Anandkumar, A. Spherical fourier neural operators: Learning stable dynamics on the sphere. In *International conference on machine learning*, pp. 2806–2823. PMLR, 2023.

Boussif, O., Bengio, Y., Benabbou, L., and Assouline, D. Magnet: Mesh agnostic neural pde solver. *Advances in Neural Information Processing Systems*, 35:31972–31985, 2022.

Challu, C., Olivares, K. G., Oreshkin, B. N., Ramirez, F. G., Canseco, M. M., and Dubrawski, A. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6989–6997, 2023.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Defferrard, M., Milani, M., Gusset, F., and Perraudin, N. Deepsphere: a graph-based spherical cnn. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1e3OlStPB.

Fathony, R., Sahu, A. K., Willmott, D., and Kolter, J. Z. Multiplicative filter networks. In *International Conference on Learning Representations*, 2020.

Fragkiadaki, K., Agrawal, P., Levine, S., and Malik, J. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

Gallagher, R., Oden, J., Taylor, C., and Zienkiewicz, O. Finite element. *Analysis: Fundamentals. Prentice Hall, Englewood Cliffs*, 1975.

Gao, Z., Shi, X., Wang, H., Zhu, Y., Wang, Y. B., Li, M., and Yeung, D.-Y. Earthformer: Exploring space-time transformers for earth system forecasting. *Advances in Neural Information Processing Systems*, 35:25390–25403, 2022.

Geneva, N. and Zabaras, N. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403: 109056, 2020.

Grattarola, D. and Vandergheynst, P. Generalised implicit neural representations. *Advances in Neural Information Processing Systems*, 35:30446–30458, 2022.

Hao, Z., Hao, J., Peng, Z., Wang, S., Yu, P. S., Wang, X., and Wang, J. Dy-hien: Dynamic evolution based deep hierarchical intention network for membership prediction. In *WSDM*, pp. 363–371, 2022.

Iakovlev, V., Heinonen, M., and Lähdesmäki, H. Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956*, 2020.

Iakovlev, V., Heinonen, M., and Lähdesmäki, H. Learning space-time continuous latent neural pdes from partially observed states. *Advances in Neural Information Processing Systems*, 36, 2024.

Kharazmi, E., Zhang, Z., and Karniadakis, G. E. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al. Learning skillful medium-range

global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, J., Sun, G., Zhao, G., and Li-wei, H. L. Robust low-rank discovery of data-driven partial differential equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 767–774, 2020a.

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

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

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Luo, X., Xu, W., Ren, Y., Yoo, S., and Nadiga, B. Continuous field reconstruction from sparse observations with implicit neural networks. *arXiv preprint arXiv:2401.11611*, 2024.

Mechee, M. S. and H Aidi, S. Generalized euler and runge-kutta methods for solving classes of fractional ordinary differential equations. *International Journal of Nonlinear Analysis and Applications*, 13(1):1737–1745, 2022.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Raonic, B., Molinaro, R., De Ryck, T., Rohner, T., Bartolucci, F., Alaifari, R., Mishra, S., and de Bézenac, E. Convolutional neural operators for robust and accurate learning of pdes. *Advances in Neural Information Processing Systems*, 36, 2024.

Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russel, T., Sanchez-Gonzalez, A., Yang, V., Carver, R., Agrawal, S., et al. Weatherbench 2: A benchmark for the next generation of data-driven global weather models. *arXiv preprint arXiv:2308.15560*, 2023.

Schober, M., Duvenaud, D. K., and Hennig, P. Probabilistic ode solvers with runge-kutta means. *Advances in neural information processing systems*, 27, 2014.

Song, S., Mukerji, T., and Hou, J. Bridging the gap between geophysics and geology with generative adversarial networks. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–11, 2021.

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

Stoll, R., Gibbs, J. A., Salesky, S. T., Anderson, W., and Calaf, M. Large-eddy simulation of the atmospheric boundary layer. *Boundary-Layer Meteorology*, 177:541–581, 2020.

Tan, C., Gao, Z., Wu, L., Xu, Y., Xia, J., Li, S., and Li, S. Z. Temporal attention unit: Towards efficient spatiotemporal predictive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18770–18782, 2023.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction. *KDD*, 2019.

Wu, H., Wang, H., Wang, K., Wang, W., Ye, C., Tao, Y., Chen, C., Hua, X.-S., and Luo, X. Prometheus: Out-of-distribution fluid dynamics modeling with disentangled graph ode. In *Proceedings of the 41st International Conference on Machine Learning*, pp. PMLR 235, Vienna, Austria, 2024a. PMLR.

Wu, H., Weng, K., Zhou, S., Huang, X., and Xiong, W. Neural manifold operators for learning the evolution of physical dynamics. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3356–3366, 2024b.

Yin, Y., Kirchmeyer, M., Franceschi, J.-Y., Rakotomamonjy, A., and Gallinari, P. Continuous pde dynamics forecasting with implicit neural representations. In *NeurIPS*, 2022.

Zheng, D., Luo, V., Wu, J., and Tenenbaum, J. B. Unsupervised learning of latent physical properties using perception-prediction networks. *arXiv preprint arXiv:1807.09244*, 2018.

A. Detailed Derivation of Theorem 3.1

In this section, we provide a step-by-step derivation for Theorem 3.1, showing how the Lipschitz continuity of the flow field and the discrete autoregressive corrections jointly yield a geometric decay in the prediction error.

A.1. Preliminaries and Setup

We consider the dynamical system

$$\frac{d}{dt}y(t) \;=\; \mathcal{F}\big(y(t),\theta\big),$$

where $\mathcal{F}$ is assumed to be L-Lipschitz continuous in y. Formally,

$$\|\mathcal{F}(y_1,\theta)-\mathcal{F}(y_2,\theta)\| \;\le\; L\,\|y_1-y_2\| \quad \text{for all} \quad y_1,y_2.$$

- True trajectory: $\mathcal{P}(t)$ is the solution to the system with initial condition $\mathcal{P}(0)=x(0)$.
- Predicted trajectory: $\mathcal{F}(t)$ is the solution of the same ODE *model* but may differ from the true trajectory due to modeling errors or parameter inaccuracies.

Define the instantaneous error as:

$$e(t) \;=\; \|\mathcal{P}(t)-\mathcal{F}(t)\|.$$

A.2. Error Growth Without Corrections

First, suppose there are no correction steps. Let

$$\delta(t) \;=\; \mathcal{P}(t) \;-\; \mathcal{F}(t).$$

Then,

$$\frac{d}{dt}\,\delta(t) \;=\; \frac{d}{dt}\,\mathcal{P}(t)-\frac{d}{dt}\,\mathcal{F}(t) \;=\; \mathcal{F}\big(\mathcal{P}(t),\theta\big) \;-\; \mathcal{F}\big(\mathcal{F}(t),\theta\big).$$

By the L-Lipschitz property,

$$\big\|\mathcal{F}(\mathcal{P}(t),\theta)-\mathcal{F}(\mathcal{F}(t),\theta)\big\| \;\le\; L\,\|\delta(t)\|.$$

Thus,

$$\frac{d}{dt}\,\|\delta(t)\| \;\le\; L\,\|\delta(t)\|.$$

Applying Gronwall's inequality from 0 to t yields

$$\|\delta(t)\| \;=\; e(t) \;\le\; e^{Lt}\,\|\delta(0)\|.$$

Hence, in the absence of corrections, the error can grow *at most* exponentially at rate L.

A.3. Incorporating Discrete Corrections

We now introduce discrete corrections at times:

$$t_1,\,t_2,\,\ldots,\,t_k,\,\ldots \quad \text{with spacing} \quad \Delta t_{\text{corr}} = t_{k+1}-t_k.$$

Right before the k-th correction, the predicted state is $y(t_k^-)$, and right after the correction, it is:

$$y(t_k^+) \;=\; y(t_k^-)+\mathcal{R}\big(y(t_k^-)\big),$$

where $\mathcal{R}$ is a *correction operator* designed to reduce the discrepancy between the prediction and the truth, with a bounded norm:

$$\|\mathcal{R}(y)\| \;\le\; \gamma.$$

Usually, $\mathcal{R}$ is chosen so as to "pull" the predicted state closer to $\mathcal{P}(t_k)$.

A.3.1. SINGLE-STEP ERROR EVOLUTION

Consider one interval between consecutive corrections, from t_{k-1}^+ to t_k^+. Suppose we already know a bound on the error at t_{k-1}^+, namely $\|e(t_{k-1}^+)\|$. We want to bound $\|e(t_k^+)\|$.

(i) From t_{k-1}^+ to t_k^-. Over the continuous-time interval $\left[t_{k-1}^+,\, t_k^-\right]$, the error follows the same Lipschitz-based growth:

$$\|e(t_k^-)\| \;\le\; e^{L\left(t_k - t_{k-1}\right)}\,\|e(t_{k-1}^+)\| \;=\; e^{L\,\Delta t_{\text{corr}}}\,\|e(t_{k-1}^+)\|.$$

(ii) Applying the correction at t_k. Immediately after the correction,

$$\|e(t_k^+)\| \;\le\; \eta\,\|e(t_k^-)\| + \|\mathcal{R}(y(t_k^-))\|.$$

If $\mathcal{R}$ effectively cancels a large portion of the discrepancy, η may be less than 1. Often one simplifies this expression into a single multiplicative factor:

$$\|e(t_k^+)\| \;\le\; \alpha\,\|e(t_{k-1}^+)\|,$$

for some constant $\alpha \in (0,1)$. The parameter α encapsulates both the exponential growth within the interval and the corrective reduction.

Putting these two parts together, we conclude that each correction step reduces the error by at least a factor of α:

$$\|e(t_k^+)\| \;\le\; \alpha\,\|e(t_{k-1}^+)\|.$$

A.4. Geometric Error Decay Over Multiple Corrections

Let $\|e(0)\| = e_0$. After K corrections, each shrinking the error by α, we get:

$$\|e(t_K^+)\| \;\le\; \alpha^K\,\|e(0)\|.$$

If t is a continuous time and $K = \lfloor t/\Delta t_{\text{corr}} \rfloor$ denotes the number of corrections applied by time t, then

$$\|e(t)\| \;\le\; \|e(t_K^+)\| \;\le\; \alpha^K\,\|e(0)\|.$$

We collect all constants (e.g., the initial error $\|e(0)\|$, the Lipschitz constant L, and any residual factors) into a single constant C. Thus, we obtain:

$$\|e(t)\| \;\le\; C\,\alpha^{\lfloor t/\Delta t_{\text{corr}} \rfloor}.$$

This completes the derivation of the geometric decay result stated in Theorem 3.1.

B. Detailed Description of Datasets

Navier-Stokes Equations. Navier-Stokes Equations (Li et al., 2020b) describe the motion of fluid substances such as liquids and gases. These equations are a set of partial differential equations that predict weather, ocean currents, water flow in a pipe, and air flow around a wing, among other phenomena. The equations arise from applying Newton's second law to fluid motion, together with the assumption that the fluid stress is the sum of a diffusing viscous term proportional to the gradient of velocity, and a pressure term. The equations are expressed as follows:

$$\begin{aligned} \rho\left(\frac{\partial u}{\partial t} + u\cdot\nabla u\right) &= -\nabla p + \nabla\cdot\tau + f, \\ \frac{\partial \rho}{\partial t} + \nabla\cdot(\rho u) &= 0, \\ \frac{\partial(\rho E)}{\partial t} + \nabla\cdot((\rho E + p)u) &= \nabla\cdot(\tau\cdot u) + \nabla\cdot(k\nabla T) + \rho f\cdot u, \end{aligned} \tag{18}$$

where u denotes the velocity field, ρ represents the density of the fluid, p is the pressure, τ is the viscous stress tensor, given by $\mu(\nabla u + (\nabla u)^T) - \frac{2}{3}\mu(\nabla\cdot u)\mathbf{I}$. E is the total energy per unit mass, $E = e + \frac{1}{2}|u|^2$, e is the internal energy per unit mass, T denotes the temperature, and k represents the thermal conductivity.

Rayleigh-Bénard Convection. Rayleigh-Bénard Convection (Wang et al., 2019) is generated using the Lattice Boltzmann Method to solve the 2-dimensional fluid thermodynamics equations for two-dimensional turbulent flow. The general form of the equations is expressed as:

$$\begin{aligned} \nabla \cdot u &= 0 \\ \frac{\partial u}{\partial t} + (u \cdot \nabla)u &= -\frac{1}{\rho_0}\nabla p + \nu \nabla^2 u + [1 - \alpha(T - T_0)]X \\ \frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla)T &= \kappa \nabla^2 T, \end{aligned} \tag{19}$$

where g is the gravitational acceleration, $\mathbf{X}$ is the acceleration due to the body-force of the fluid parcel, ρ_0 is the relative density, T represents temperature, T_0 is the average temperature, α denotes the coefficient of thermal expansion, and κ is the thermal conductivity coefficient. The simulation parameters for the dataset are as follows: Prandtl number = 0.71, Rayleigh number = 2.5×10^8, and the maximum Mach number = 0.1.

Prometheus. Prometheus (Wu et al., 2024a) is a large-scale, out-of-distribution (OOD) fluid dynamics dataset designed for the development and benchmarking of machine learning models, particularly those that predict fluid dynamics under varying environmental conditions. This dataset includes simulations of tunnel and pool fires (representated as Prometheus-T and Prometheus-P in experiments), encompassing a wide range of fire dynamics scenarios modeled using fire dynamics simulators that solve the Navier-Stokes equations. Key features of the dataset include 25 different environmental settings with variations in parameters such as Heat Release Rate (HRR) and ventilation speeds. In total, the Prometheus dataset encompasses 4.8 TB of raw data, which is compressed to 340 GB. It not only enhances the research on fluid dynamics modeling but also aids in the development of models capable of handling complex, real-world scenarios in safety-critical applications like fire safety management and emergency response planning.

WeatherBench. WeatherBench (Rasp et al., 2023) is a benchmark dataset designed for the evaluation and comparison of machine learning models in the context of medium-range weather forecasting. It is intended to facilitate the development of data-driven models that can improve weather prediction, particularly in the range from 1 to 14 days ahead. The dataset consists of historical weather data from multiple atmospheric variables, including temperature, pressure, humidity, wind speed, and geopotential height, at various global locations. The data is derived from the ERA5 reanalysis, which provides hourly estimates of the atmosphere's state at a resolution of 31 km for the period from 1950 to present. Its primary goal is to serve as a benchmarking tool to compare the performance of machine learning-based models against traditional numerical weather prediction methods. By focusing on data-driven techniques, it aims to push forward the development of models that can predict weather patterns in an interpretable and scalable manner, and thus contribute to improving operational weather forecasting systems.

C. Details of Implementation

To ensure fairness, we conducted all experiments on an NVIDIA-A100 GPU using the MSE loss over 200 epochs. We used Adam optimizer with a learning rate of 10^{-3} for training. The batch size was set to 16.

C.1. Model hyper-parameters

The core architecture of the model consists of three main modules: the spatial information encoder, the multi-scale graph ODE module, and the neural auto-regressive correction module. The spatial information encoder employs Multiplicative Filter Networks to fuse partial observations with spatial coordinate information and encode them into customized geometric grids. For regular arranged datasets, the customized grid is set to the general resolution. For irregular arranged datasets, it is uniformly set to 128×128. The MFN is configured with 5 layers and uses ReLU as the activation function to ensure efficient feature extraction. The hidden dimension is set to 128. Then the multi-scale graph ODE module utilizes a Runge-Kutta ode solver for numerical integration, with a time step size of 0.25 to ensure accurate modeling of temporal continuity. Further, the neural auto-regressive correction module performs corrections per integer time step. For this module, the Conv2d layer is downsampled to half the resolution, while the UpConv2d layer restores the grid to the original resolution. The parallel GroupConv2d operations are implemented with filter sizes of 3×3, 5×5, and 7×7. For inference, the correction weight λ is set to 0.5 to balance correction strength and model stability. Finally, in the decoder, we use a single step Gabor filter

transformation to initial the features of query coordinates, and perform a 3-layers message-passing update to obtain the corresponding predictions.

C.2. Baseline implementation

MAgNet (Boussif et al., 2022). We utilize the official implementation of MAgNet, utilizing a graph neural network variant of the model. The configuration involves five message-passing steps. The architecture of all MLPs includes four layers, with each layer containing 128 neurons. Additionally, we set the dimensionality of the latent state at 128.

DINo (Yin et al., 2022). We utilize the official implementation of DINo. Specifically, the encoder features an MLP, comprising three hidden layers with 512 neurons each, and Swish non-linearities. The dimension of each hidden layer is set to 100. Similarly, the dynamics function is realized through an MLP, which also includes three hidden layers, each containing 512 neurons and employing Swish activation function. The decoder is constructed with three layers, each with a capacity of 64 channels.

ContiPDE (Steeven et al., 2024). ContiPDE formulates the task as a double observation problem. It utilizes recurrent GNNs to roll out predictions of anchor states from the IC, and employs spatio-temporal attention observer to estimate the state at the query position from these anchor states. First, it utilize a two-layered MLP with 128 neurons, with Swish activation functions to encode features form sparse observations. Further, it uses a two-layered gated recurrent unit with a hidden vector of size 128, and a two-layered MLP with 128 neurons activated by the Swish function to realize the recurrent GNNs. Finally, it employs multi-head attention mechanism to decode and utilizes multi-head attention mechanism to realize continuous query.

D. Noise Disturbance Robustness.

To evaluate the robustness of our model, we present the effects of observational noise on its performance and compare these results with those of other models. We quantify the noise using the channel-specific standard deviation tailored to the dataset and have trained the model under various noise intensities (noise ratios set at 1%, 5%, 10%, 15%, and 20%). Experiments are conducted on and Navier-Stokes and Prometheus datasets. Observations from Figure 5 reveal that the proposed `COPS` effectively maintains its performance with noise ratio below 5%, and demonstrates significant advantages over other baseline models when noise ratio increases over 10%. In contrast, we observe a more pronounced performance degradation in the two interpolation-based baseline models as noise ratio raises over 10%, which indicates their weaker robustness against noise interference.

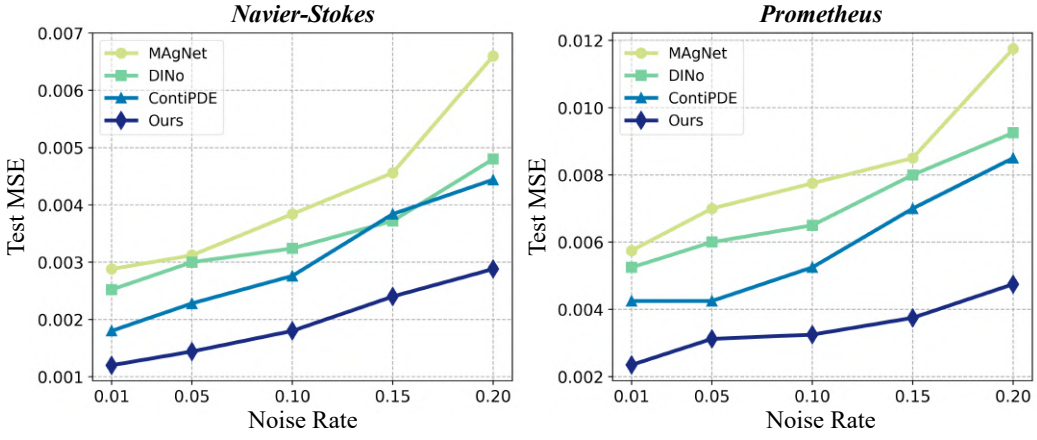

Figure 5. Performance with regard to noise disturbance.