

Sparse Diffusion Autoencoder for Test-time Adapting Prediction of Complex Systems

Jingwen Cheng*

Department of Electronic Engineering
Tsinghua University
Beijing, China

Ruikun Li*

Shenzhen International Graduate School
Tsinghua University
Shenzhen, China

Huandong Wang†

Department of Electronic Engineering
BNRist, Tsinghua University
Beijing, China

Yong Li

Department of Electronic Engineering
BNRist, Tsinghua University
Beijing, China

Abstract

Predicting the behavior of complex systems is critical in many scientific and engineering domains, and hinges on the models ability to capture their underlying dynamics. Existing methods encode the intrinsic dynamics of high-dimensional observations through latent representations and predict autoregressively. However, these latent representations lose the inherent spatial structure of spatiotemporal dynamics, leading to the predictor’s inability to effectively model spatial interactions and neglect emerging dynamics during long-term prediction. In this work, we propose SparseDiff, introducing a test-time adaptation strategy to dynamically update the encoding scheme to accommodate emergent spatiotemporal structures during the long-term evolution of the system. Specifically, we first design a codebook-based sparse encoder, which coarsens the continuous spatial domain into a sparse graph topology. Then, we employ a graph neural ordinary differential equation to model the dynamics and guide a diffusion decoder for reconstruction. SparseDiff autoregressively predicts the spatiotemporal evolution and adjust the sparse topological structure to adapt to emergent spatiotemporal patterns by adaptive re-encoding. Extensive evaluations on representative systems demonstrate that SparseDiff achieves an average prediction error reduction of 49.99% compared to baselines, requiring only 1% of the spatial resolution.

1 Introduction

The dynamics of complex systems are driven by the nonlinear interactions and co-evolution of numerous components, giving rise to rich emergent spatiotemporal structures, as seen in fluid dynamics [1], climate science [2], and molecular dynamics [3]. Accurate long-term prediction of these systems is crucial for many real-world applications [4, 5]. However, complex systems are typically observed in high-dimensional spaces with unknown intrinsic dy-

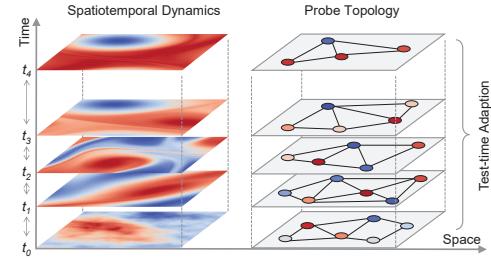


Figure 1: Adapting probe topologies.

*Equal contribution (see § 7).

†Corresponding author (wanghuandong@tsinghua.edu.cn)

namics, making high-fidelity, long-term prediction a challenging and computationally expensive problem [6, 7, 8].

Data-driven methods offer an equation-free and computationally friendly approach. The core idea is to build reduced-order models of high-dimensional observations using encoder-decoders, which project observed states to a low-dimensional latent space for prediction, thereby improving computational efficiency [6, 9, 10, 11, 12]. However, most existing work employs parameterized neural network encoders that implicitly encode spatial structure by compressing observational spatial information into a latent representation vector. This conversion of spatially-correlated data into latent vectors hinders predictors from effectively understanding and modeling the system’s inherent spatial interactions and structural relationships [5]. For instance, in fluid convection, heat drives fluid to form rising plumes and sinking cold flows. These specific spatial structures, their relative positions, and interactions are critical for understanding and predicting convection patterns [13]. Considering that spatial interaction patterns and emergent structures continuously change during long-term evolution, losing such information progressively amplifies the misalignment between the encoder-decoder and predictor in long-term predictions. This leads to a core question: Can we construct a reduced-order model that preserves crucial spatial structure and continuously adapts to newly emerging dynamics patterns during prediction?

This work is inspired by a recent finding that the system state in the entire spatial domain can be effectively reconstructed from sparse observation points [14, 15, 16]. The data disparity between sparse points and the full spatial grid strongly suggests a promising encoding paradigm: aggregating the full-space dynamic information onto a much smaller set of sparse probes [17]. These probes and their spatial topology constitute the dynamical skeleton of the spatiotemporal dynamics, as shown in Figure 1. In this paradigm, the model is capable of adapting to the system’s latest spatiotemporal patterns during long-term predictions by dynamically adjusting probe positions and topology based on the re-encoded predicted states [6, 18]. Nevertheless, achieving effective probe-based aggregation and adaptive updating in practice faces two key challenges: 1) A lack of theoretical understanding for aggregating rich spatiotemporal dynamics from a continuous domain into a small set of discrete probes; 2) Frequent re-encoding during long-term prediction introduces significant computational overhead and potential accumulation of prediction errors.

To address these challenges, we propose a novel Sparse Diffusion Autoencoder, SparseDiff, consisting of a codebook-based sparse encoder and an unconditional diffusion decoder. The sparse encoder learns a representative pattern codebook from historical spatiotemporal trajectories, enabling it to dynamically aggregate and project full-space observational data onto a spatial topology formed by sparse probes, thus constructing a reduced-order model. On this probe topology, we model spatiotemporal dynamics using a graph neural ordinary differential equation and explicitly introduce a diffusion term to capture spatial interactions and information propagation among probes. Finally, we pad the probe predictions to the full spatial domain to serve as the initial state for the diffusion process, enabling rapid reconstruction of the full spatiotemporal field. SparseDiff’s key innovation and strength lie in its test-time adaptation: during long-term prediction, it utilizes the learned codebook to re-encode the latest prediction, dynamically adjusting and constructing a probe topology better matched to the current system state to continuously adapt to emerging spatiotemporal patterns. Experimental evaluation on simulated and real-world systems demonstrates that SparseDiff outperforms baselines by over 49.99% in long-term predictions, and reliably predicts full-space spatiotemporal dynamics using less than 1% of grid points as probes.

The highlights of this work are summarized as follows:

- We propose SparseDiff, a novel autoencoder that uses sparse probes to construct a reduced-order model that effectively preserves the spatial structure of spatiotemporal dynamics.
- We introduce a test-time adaptation strategy via re-encoding, allowing the model to dynamically adapt its probe-based representation to emerging dynamics patterns, significantly improving long-term prediction accuracy and computational efficiency.
- Experiments demonstrate that SparseDiff significantly outperforms baselines in long-term prediction accuracy and achieves high efficiency by reliably predicting full-space dynamics using only approximately 1% of grid points as probes. Our code is open-source: <https://github.com/tsinghua-fib-lab/SparseDiff>.

2 Preliminary

2.1 Problem Definition

In this work, we focus on spatiotemporal dynamics $\frac{du}{dt} = f(u, x, t, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \dots)$ on a 2D regular spatial domain $x \subset \mathbb{R}^2$. Such systems can often be formalized as time-dependent partial differential equations, for example, the Navier-Stokes equations, which include a Laplacian operator term acting on the spatial domain [19]. Without loss of generality, we address the problem of data-driven modeling of system dynamics from historical evolving trajectories $U = \{u(x, t)\}^T$, thereby extending our focus to real-world systems like climate dynamics where explicit equations are unknown. In practice, the spatial domain is discretized into an $h * w$ grid. Consequently, the full-domain observation state at each timestamp is represented as a tensor $u \in \mathbb{R}^{h \times w}$. Given a historical observation trajectory of *lookback* steps, we predict multiple future steps in an autoregressive manner, forecasting one step at a time.

2.2 Guided Diffusion for Sparse Reconstruction

Diffusion methods have recently been shown to reliably reconstruct the full spatial domain’s system state, guided by sparse observations [16, 15, 14]. Diffusion models aim to learn a probabilistic mapping from a simple prior distribution, such as a standard Gaussian to a complex target data distribution [20, 21, 22]. This is achieved by defining a forward diffusion process that gradually adds noise to data instances and training a reverse process to sequentially denoise them. We represent the original data point as x_0 . The forward stage transforms x_0 into a noisy version x_n over n steps, governed by the equation $x_n = \sqrt{\bar{\alpha}_n}x_0 + \sqrt{1 - \bar{\alpha}_n}\epsilon_n$, ϵ_n and $\{\bar{\alpha}_n\}$ represent the Gaussian noise and noise schedule [23], respectively. The learned reverse diffusion process models the transition from noise back to data through a sequence of conditional distributions given by

$$p_\theta(x_{n-1}|x_n) := \mathcal{N}(x_{n-1}; \mu_\theta(x_n, n), \sigma_n^2 \mathbf{I}), \quad (1)$$

where $\mu_\theta = \frac{1}{\sqrt{\bar{\alpha}_n}}(x_n - \frac{1 - \bar{\alpha}_n}{\sqrt{1 - \bar{\alpha}_n}}\epsilon_\theta(x_n, n))$ and $\{\sigma_n\}$ are step dependent constants. The term ϵ_θ represents the model’s prediction of the added noise, typically implemented using a parameterized neural network architecture like a UNet or Transformer. The optimization of this network’s parameters is performed by minimizing the objective function [23]

$$L_n = \mathbb{E}_{n, \epsilon_n, x_0} \|\epsilon_n - \epsilon_\theta(\sqrt{\bar{\alpha}_n}x_0 + \sqrt{1 - \bar{\alpha}_n}\epsilon_n, n)\|^2. \quad (2)$$

This loss function is derived from the negative log-likelihood $\mathbb{E}_{x_0 \sim q(x_0)}[-p_\theta(x_0)]$. Once trained, the diffusion model progressively denoises from Gaussian noise to yield high-fidelity data samples. To guide diffusion in reconstructing the full spatial domain state from K sparse observations $\mathcal{M}(x_K)$, Bayes’ rule guides the diffusion gradient direction [24] to be

$$\nabla_{x_n} \log p(x_n | \mathcal{M}) \approx -\frac{1}{\sqrt{1 - \bar{\alpha}_n}}\epsilon_\theta - \zeta \nabla_{x_n} \|y - \mathcal{M}(x_K)\|_2^2, \quad (3)$$

where y represents the noise values at the sparse observation points and $\zeta = 1/\sigma^2$.

3 Methodology

In this section, we first introduce the proposed sparse diffusion autoencoder to discover the skeleton of spatiotemporal dynamics, namely the probe topology. Subsequently, we design a diffusion graph neural ordinary differential equation on the probe topology to model spatiotemporal dynamics. Finally, we propose the test-time adaptation strategy to dynamically sense emerging spatiotemporal patterns in long-term prediction. The overall framework is illustrated in Figure 2.

3.1 Sparse Diffusion Autoencoder

We introduce our approach for discovering the dynamical skeleton of complex systems, utilizing a codebook-based sparse encoder and a guided diffusion decoder.

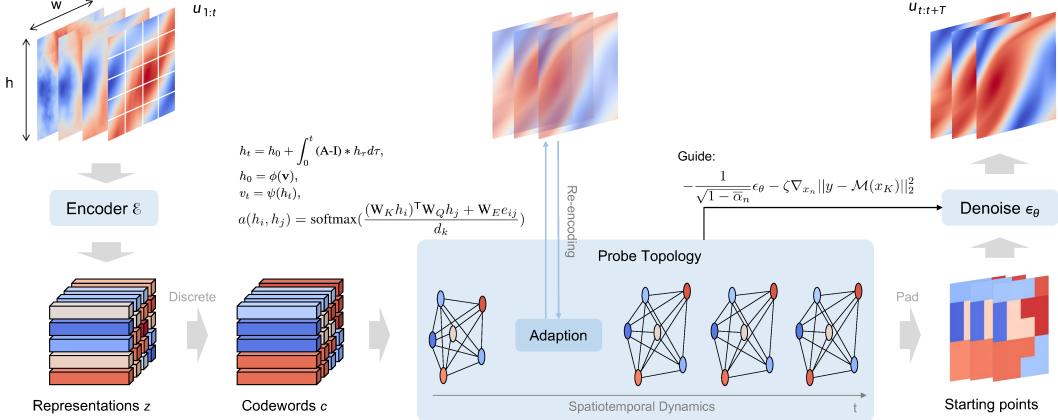


Figure 2: Overall framework of proposed Sparse Diffusion Autoencoder.

3.1.1 Codebook-based Sparse Encoder

To discover the skeleton of spatiotemporal dynamics on the spatial domain, we first identify and record the rich dynamical patterns inherent in complex systems. Specifically, we maintain a codebook C containing K codewords for spatiotemporal dynamics, where each codeword $c \in \mathbb{R}^d$ is a d -dimensional learnable vector. For a given historical observation $\mathbf{u} \in \mathbb{R}^{t \times h \times w}$, we employ an encoder \mathcal{E} to encode along the time dimension, yielding a d -dimensional representation vector $z \in \mathbb{R}^{d \times h \times w}$ for the full-domain states. We replace the representation vector z with the most similar codeword c , which serves as the decoder's input. The decoder then reconstructs the original observation $\hat{\mathbf{u}}$. The encoder-decoder and the codebook are trained by minimizing the objective function [25]

$$L = \text{logp}(\mathbf{u}|c) + \|\text{sg}[\mathcal{E}(\mathbf{u})] - c\|_2^2 + \beta \|\mathcal{E}(\mathbf{u}) - \text{sg}[c]\|_2^2, \quad (4)$$

where sg indicates gradient detachment. Through this process, we effectively record the rich spatiotemporal dynamical patterns from the observation trajectories within the pretrained codebook.

In the inference stage, we first discretize the historical observation u into a set of k hit codewords $\{c_i\}_{i=1}^k$ using pretrained encoder \mathcal{E} and codebook C . We define the governing region of each hit codeword c_i as the set of spatial grid points that are mapped to it, denoted as u_{c_i} . Consequently, the full spatial domain is divided into k region types by the k hit codewords (a single region type might consist of dispersed patches). These k regions represent k local-scale spatiotemporal units. Therefore, we accordingly construct probe set $\mathcal{V} = \{v_i \in \mathbb{R}^l\}_{i=1}^k$ to represent them, where l is the lookback window size.

For probe v_i , we select one spatial grid point from its corresponding codeword's governing region u_{c_i} as its coordinate. We use random selection here instead of averaging, ensuring that the probe falls within patches even if the patches of its region are spatially dispersed. Subsequently, we average the historical observation sequences of all grid points within u_{c_i} to aggregate them as probe states $\mathbf{v}_i \in \mathbb{R}^l$. Finally, we connect all probes to construct the topological structure $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, and assign different weights $e_{ij} \in E$ to each edge to characterize the spatial association strength. Specifically, edge weights e_{ij} quantify the spatial association strength from probe v_i to probe v_j . We consider the governing region u_{c_i} associated with v_i and spatial neighborhood of every grid point within u_{c_i} . We count how many grid points belonging to v_j 's governing region u_{c_j} appear within the combined neighborhoods of all points in u_{c_i} . This count is normalized along all probes to determine the edge weight e_{ij} . Thereby, we obtain the probe topology \mathcal{G} of the spatiotemporal dynamics.

3.1.2 Guided Diffusion Decoder

We pretrain an unconditional diffusion model to capture the spatiotemporal patterns of the system state u . Given the set of probes \mathcal{V} , we use these probe values v_i as sparse observations to guide the reconstruction by the diffusion decoder. Existing works [16, 15, 14] typically directly substitute probe values as the sparse observation $\mathcal{M}(x_K)$ in Equation 3 to guide reconstruction. However, our approach differs in that we guide the governing regions u_{c_i} corresponding to the probes. That is, our probes represent the dynamical information of the entire governing region, not just single grid points. Therefore, we use probe value v_i to guide diffusion in reconstructing its governing region.

Specifically, we fill the probe value v_i to each grid point within its governing region u_{c_i} , serving as $\mathcal{M}(x_K)$ in Equation 3 to guide reconstruction. Furthermore, considering the computationally intensive of the denoising process, we use the filled full-space state instead of Gaussian noise as the starting point for denoising to accelerate the diffusion sampling process.

3.2 Probe-graph Diffusive Predictor

Given the probe topology \mathcal{G} and probe states \mathbf{v}_i , we now predict the spatiotemporal dynamics of the system. Specifically, we model this process as a neural diffusion process on the graph [26]

$$\begin{aligned} h_t &= h_0 + \int_0^t (\mathbf{A} \cdot \mathbf{I}) * h_\tau d\tau, \\ h_0 &= \phi(\mathbf{v}), \\ v_t &= \psi(h_t), \end{aligned}$$

where h_t refers to the representation at time t , ϕ and ψ are the parameterized encoder and decoders, and \mathbf{A} is the time-varying diffusion coefficient between probes, calculated by a parameterized attention network. To explicitly model the spatial association strength between probes, we introduce the edge weights into the calculation of the diffusion coefficient as

$$a(h_i, h_j) = \text{softmax}\left(\frac{(\mathbf{W}_K h_i)^\top \mathbf{W}_Q h_j + \mathbf{W}_E e_{ij}}{d_k}\right),$$

where h_i denotes the representation of the i th probe, \mathbf{W}_Q , \mathbf{W}_K , and \mathbf{W}_E are learnable matrices. We employ a multi-head attention mechanism $\mathbf{A} = \frac{1}{k} \sum_k \mathbf{A}^k$ to capture complex dynamical mechanisms.

3.3 Test-time Adapting Prediction

During long-term prediction, complex system dynamics exhibit continuous emergence of spatiotemporal patterns, implying time-varying dynamical spatial interactions. Therefore, we continuously update the predictor during testing to adapt to the latest spatiotemporal dynamics patterns. For a T -step prediction, the prediction window is divided into N sub-windows $\{w_n\}^N$. We assume each window possesses specific spatial interactions, with a corresponding probe topology sequence $\{\mathcal{G}_{w_n}\}^N$. We update the probe topology at the beginning of each window through re-encoding. Specifically, at a window transition time, we decode the probe states back to full space. Following the method in Section 3.1.1, we then re-encode the probe topology \mathcal{G} for the latest dynamics within the current window using encoder \mathcal{E} and codebook C . This topology is updated again at the next transition time.

We design a dynamic update strategy to dynamically determine the window transition time. Instead of using fixed-length sub-windows, we continuously monitor whether the recent evolution of each probe remains well-aligned with its assigned codeword in the latent space of the encoder \mathcal{E} . Specifically, after each re-encoding, we obtain k hit codewords $\{c_i\}_{i=1}^k$, where each $c_i \in \mathbb{R}^d$ corresponds to a governing region. For each codeword, a probe v_i is randomly selected to represent its region and forms a probe graph \mathcal{G} for prediction. At each prediction step, we compute the latent consistency score χ_t by encoding the past T -step trajectory of each probe v_i and evaluating its cosine similarity with the associated codeword c_i :

$$\chi_t = \frac{1}{k} \sum_{i=1}^k \cos(\mathcal{E}(v_i^{t-T:t}), c_i).$$

When χ_t drops below a predefined threshold τ , it indicates that many probe representations have drifted away from their original latent codewords, suggesting a mismatch between the current probe partition and the evolving system dynamics. In response, we decode the probe graph of the past T steps back to the full space using the diffusion decoder and recompute the probe topology via re-encoding. This strategy enables the model to adaptively reallocate codewords and update the graph structure in accordance with the varying speed and patterns of spatiotemporal evolution.

Table 1: Average performance of the trajectories predicted from different initial conditions with standard deviation from 10 runs. The best results are highlighted in bold.

Systems	Lambda-Omega		Navier-Stokes		SwiftHohenberg		Cylinder-Flow		Real-world	
	RMSE $\times 10^{-2}$	SSIM $\times 10^{-1}$	RMSE $\times 10^{-2}$	SSIM $\times 10^{-1}$	RMSE $\times 10^{-2}$	SSIM $\times 10^{-1}$	RMSE $\times 10^{-2}$	SSIM $\times 10^{-1}$	RMSE $\times 10^{-2}$	SSIM $\times 10^{-1}$
ConvLSTM	5.613 \pm 0.557	9.224 \pm 0.125	30.674 \pm 2.704	4.696 \pm 0.083	9.556 \pm 0.059	9.399 \pm 0.006	11.098 \pm 0.330	5.265 \pm 0.144	13.682 \pm 2.777	7.288 \pm 1.195
FNO	5.138 \pm 0.557	9.170 \pm 0.514	15.992 \pm 1.062	7.262 \pm 1.642	19.693 \pm 5.591	9.614 \pm 0.009	19.854 \pm 25.248	7.583 \pm 0.525	12.639 \pm 3.878	6.775 \pm 1.368
UNet	8.324 \pm 0.497	8.014 \pm 0.393	16.277 \pm 1.582	7.092 \pm 0.837	14.881 \pm 2.789	9.206 \pm 0.225	21.349 \pm 4.903	6.203 \pm 0.196	13.928 \pm 2.855	6.328 \pm 0.974
G-LED	6.506 \pm 0.395	8.992 \pm 0.396	12.334 \pm 0.485	8.095 \pm 0.963	8.214 \pm 0.937	9.572 \pm 0.211	10.021 \pm 0.873	7.059 \pm 0.291	10.304 \pm 2.009	6.768 \pm 0.539
Ours	2.912 \pm 0.187	9.601 \pm 0.218	11.130 \pm 3.241	8.492 \pm 0.793	7.628 \pm 3.102	9.675 \pm 0.192	8.544 \pm 0.239	7.392 \pm 0.171	7.957 \pm 1.207	7.781 \pm 1.034
	53.56%	9.05%	38.64%	28.63%	57.42%	1.89%	62.67%	17.39%	37.86%	15.51%

4 Experiments

In this section, we validate the accuracy and efficiency of SparseDiff on simulated PDE systems and real-world datasets. Furthermore, we evaluate its robustness and generalization ability and examine the specific contribution of its components through ablation studies.

4.1 Experimental Setup

We conduct experimental validation on four PDE systems, including: 1) Lambda-Omega; 2) Navier-Stokes; 3) Cylinder Flow; and 4) SwiftHohenberg systems. They encompass complex diffusion effects, convection terms, and higher-order spatial interaction terms, among other nonlinear dynamic components. Additionally, we also evaluate SparseDiff’s performance in real-world applications on an open-source climate record dataset [27]. All models are trained on trajectories with different initial conditions and predict long-term (more than 100 steps) on new trajectories. Details on the equations, data generation, and training settings are in Appendix A.

Baselines We compare our method against a set of state-of-the-art spatiotemporal forecasting models, including operator learning methods, recurrent architectures, and generative frameworks. Specifically, we consider Fourier Neural Operator (FNO) [28], ConvLSTM [29], UNet [30], and G-LED [31]. A detailed description of each baseline is provided in Appendix C.

4.2 Main Results

PDE systems Table 1 shows the long-term prediction performance of all models. SparseDiff achieves the optimal prediction quality in almost all scenarios. This indicates that SparseDiff’s test-time adaptation enhances long-term prediction by timely sensing of new spatio-temporal dynamics. Furthermore, compared to other baselines, another characteristic of SparseDiff is that it performs dynamic prediction on the probe topology. Other methods predict in the original or uniformly downsampled grid space and, unlike SparseDiff, fail to discover the spatial interactions of complex systems and structure them into an explicit topology to enhance the predictor’s representational capacity.

Real-world dataset We use the SEVIR dataset [27] for real-world weather forecasting. Specifically, we select the GOES-16 Channel 09 ($6.9\mu\text{m}$) infrared imagery ir069, which captures mid-level water vapor and is widely used for storm tracking. Each frame covers a 384×384 km area at 2 km resolution, yielding a 192×192 grid, with a temporal resolution of 5 minutes. Experimental results are shown in Figure 3, where SparseDiff achieves the most faithful prediction of complex climate patterns with the lowest error. This indicates that the proposed probe-based encoding scheme is of great value for applications in real-world complex systems, especially in large-scale observational data scenarios in geoscience. SparseDiff en-

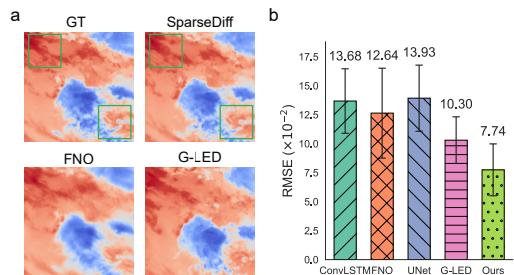


Figure 3: Real-world dataset. (a) Prediction visualization of different models. (b) RMSE comparison of different models.

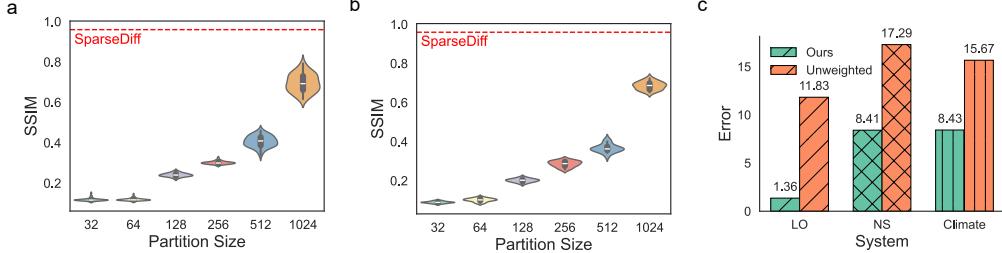


Figure 4: Ablation studies. Prediction performance with (a) uniform and (b) random probe selection. (c) Impact of probe topology edge weights on prediction.

codes high-dimensional observations into a sparse probe topology, thus reducing computational overhead while ensuring accuracy.

4.3 Ablation Study

We capture a compact low-dimensional skeleton of spatiotemporal dynamics through codebook learning, thereby improving prediction performance. The contrasting version is uniform down-sampling or random selection of probes. We also establish heuristic calculation rules for the edge weights of the probe topology. The contrasting version is an unweighted graph (where all weights are set to 1). In the following, we ablate these two key components.

Selection of Probes Probes are the most critical components in the SparseDiff framework, as they serve as the information carriers that represent the coarse-grained structure of the spatiotemporal dynamics. Their selection fundamentally determines the prediction quality of the entire model. Figures 4a and b illustrate the prediction performance of SparseDiff on the SwiftHohenberg system when the probes are selected via spatially uniform and random sampling, respectively. We observe that even when the number of probes is increased to 1024, these two variants still perform significantly worse than our method. In contrast, the original version achieves high-quality predictions with as few as 150 probes. This highlights that our proposed codebook-based sparse encoder not only compresses the spatial domain effectively but also selects informative and dynamically representative probes. It demonstrates a strong ability to extract the low-dimensional spatiotemporal skeleton where the intrinsic dynamics reside.

Probe-Graph Edge Weights When the edge weight feature is disabled, the edge-feature-aware attention mechanism described in Section 3.2 degenerates into standard inter-node attention that treats all probe connections equally. Without incorporating the spatial association strength e_{ij} , the model loses the ability to capture relative distances and directional relationships between probes, which are essential in spatiotemporal systems. As shown in Figure 4c, this leads to a notable performance drop on the Navier-Stokes system, where accurate modeling of spatial interactions is critical for capturing fine-grained dynamics.

4.4 Robustness

Here, we evaluate the robustness of SparseDiff to data observation noise and codebook size settings. Specifically, we first test the prediction performance with different preset codebook sizes on the Navier-Stokes system. The codebook size is the upper limit on the number of activated codewords. It determines the maximum number of probes SparseDiff can select to perceive spatiotemporal dynamics. Experimental results are shown in Figure 5a, where the prediction accuracy (SSIM) converges after the codebook size reaches 150. For the Navier-Stokes system with 128E128 resolution, this is

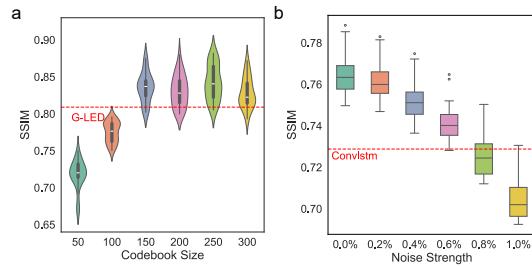


Figure 5: Robustness experiments. (a) Impact of codebook size on SparseDiff’s performance on the Navier-Stokes system. (b) Impact of the noise on SparseDiff’s performance on the real climate dataset.

quite a small number. This indicates that SparseDiff utilizes probes efficiently and achieves superior prediction with very low computational overhead.

Next, we examine SparseDiff's robustness to real-world noise. We apply Gaussian noise with varying percentage intensity to a real-world climate dataset (relative to the original data amplitude). Experimental results are shown in Figure 5b, where SparseDiff's performance deteriorates as the noise increases, but at moderate noise levels, it still outperforms the baseline without noise.

4.5 Generalization

We examine SparseDiff’s out-of-distribution generalization ability. Specifically, using the Cylinder Flow system as the experimental subject, we evaluate SparseDiff’s ability to predict flows with Reynolds numbers not included in the training set. For turbulent systems, the Reynolds number influences the viscous coefficient, thereby affecting collision frequency. For fluids with high Reynolds numbers, the flow exhibits chaotic tendencies and is thus difficult to predict over long periods. We train SparseDiff on Reynolds numbers less than 500 and predict on larger Reynolds numbers. Experimental results are shown in Figure 6, where the baseline shows rapid performance degradation in out-of-distribution scenarios, whereas SparseDiff’s performance consistently outperforms the baseline and exhibits fluctuations. The reason may be that a rich set of spatiotemporal dynamic patterns is recorded in SparseDiff’s pretrained codebook. Although there are differences in the turbulent dynamic behaviors at different Reynolds numbers, local small-scale dynamic patterns share similarities and are thus recognized by SparseDiff and accurately generalized.

4.6 Trade-off between Accuracy & Efficiency

Here, we analyze how SparseDiff trades off between accuracy and efficiency. Specifically, during testing, SparseDiff can adjust the probe topology through re-encoding, thus constructing the most suitable form for newly emerging spatiotemporal dynamics. Frequent adjustments allow for sufficient perception of real-time dynamics, but the corresponding computational overhead increases. We conducted tests at different update intervals on the Navier-Stokes system, and the results are shown in Figure 7, where *Rollout Step* refers to the time index in the autoregressive prediction sequence, and each step corresponds to one forward prediction in the rollout process. According to Figure 7a, accuracy and time overhead exhibit a negative correlation.

We also compare with the best-performing baseline, G-LED. It can enhance long-term prediction accuracy by decoding and re-encoding at intervals, but its accuracy ceiling is lower than SparseDiff, and its time overhead is much greater than SparseDiff. Our adaptive re-encoding strategy proposed in Section 3.3 achieves a balance between accuracy and efficiency, as shown by the red star in Figure 7a.

4.7 Comparison of Graph Construction Schemes

To validate our proposed region-aware edge weighting scheme, we compare it against two primary baseline categories: (a) Learnable Graph Construction: The Graph Kernel Network (GKN) [32], a method designed for learnable graph construction; and (b) Heuristic Alternatives: Using the same

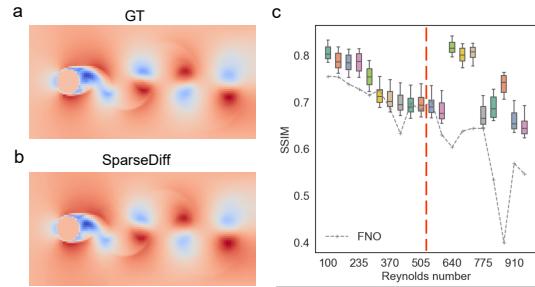


Figure 6: Generalization experiments. Visualized (a) ground truth and (b) SparseDiff prediction of turbulent at $Re = 660$. (c) SSIM as a function of Reynolds number.

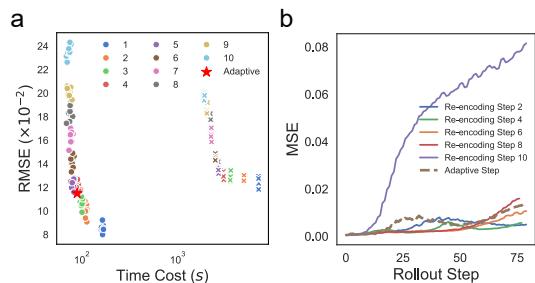


Figure 7: Trade-off of accuracy & efficiency. Circles represent SparseDiff, and crosses represent GLED.

GRAND predictor but replacing our learned e_{ij} with static graph structures, including k-nearest neighbors (kNN) based on Euclidean coordinates ($k = 20\%$ of probe count), and spectral clustering followed by kNN in the embedding space.

We report the average RMSE over a 100-step autoregressive prediction on the LO system. As shown in Table 2, our region-aware edge scheme yields significantly better performance than both the learnable GKN and the heuristic variants. This demonstrates that explicitly encoding region-level adjacency improves the expressiveness of the model for probe-based dynamics.

Table 2: Comparison of different graph construction methods.

Method	RMSE $\times 10^{-2}$
Ours	2.873
GKN	4.210
GRAND + kNN	3.287
GRAND + spectral	3.019

5 Related Work

5.1 Encoding Dynamics of Complex Systems

Discovering the low-dimensional latent space containing the intrinsic dynamics of complex systems represents a core challenge in long-term prediction. To accelerate spatiotemporal dynamical system prediction, some earlier studies focus on approximating solutions on coarse grids. Lee et al. [7] utilize Gaussian processes and diffusion maps to coarse-grain high-resolution microscopic observations into coarse-scale PDEs. Bar-Sinai et al. [33] employ uniform grid downsampling of the continuous spatial domain for specific nonlinear partial differential equations. This coarse-grid approach has inspired several recent works [31, 34]. With advancements in representation learning, autoencoders have been employed to data-drivenly uncover the low-dimensional latent space of high-dimensional observations [6, 35]. Prediction models such as recurrent neural networks [6] and neural operators [9] operate directly within the low-dimensional space constructed by autoencoders. To ensure the latent space aligns with physical intuition, existing research suggests integrating autoencoders with physical priors [36]. Li et al. [11] restrict the output semantics of the encoder by specifying timescales and intrinsic dimensions. Wu et al. [10] combine delay embeddings with feature embeddings to discover the low-dimensional manifold of high-dimensional observations. Additionally, some work [37] learns the latent coordinates of partial differential equations through autoencoders to reveal governing equations. In contrast to these methods, our proposed probe topology adaptively senses spatial structure, without enforcing a regular grid. This explicitly preserves the tight spatial correlations of spatiotemporal dynamics with a high compression ratio.

5.2 Generative Modeling for Complex Systems

Diffusion models have performed remarkably well in generative tasks, with their significant achievements in video synthesis [38, 39] and time series modeling [40, 41] inspiring numerous studies in complex system dynamics prediction. Some works [42, 15] have utilized large-scale pre-trained diffusion models to reconstruct high-fidelity data from lower-fidelity samples or sparse measurement data. Physics knowledge is also integrated. For example, known partial differential equations provide physical constraints (PDE loss) for the denoising step, improving accuracy [43]. Beyond reconstruction, diffusion models have also been applied to generate specific dynamical data. Li et al. [44] propose a machine learning approach for generating single-particle trajectory data in high Reynolds number three-dimensional turbulence. Lienen et al. [13] treat turbulence simulation itself as a generative task, employing diffusion models to capture the distribution of turbulence induced by unseen objects and generate high-quality samples for downstream applications. Other strategies focus on directly embedding dynamics or prediction processes into the diffusion mechanism. Cachay et al. [45] align the temporal evolution axis of spatiotemporal dynamics with diffusion process steps, replacing noise injection with temporal interpolation and the denoising operation with prediction. G-LED [31], a more direct approach, incorporates system states as prediction targets and utilizes predicted future frames as conditions to guide the diffusion model in reconstructing the original high-fidelity states. Li et al. [5] embed multiscale features as conditioning for Diffusion modeling, while [46] dynamically adjust the number of denoising steps based on dynamical timestamps. Compared to these methods, we introduce a novel sparse encoder working in tandem with a diffusion decoder. This combination not only reveals the low-dimensional spatial structure of long-term dynamics but also enables test-time adaptation to emerging spatiotemporal dynamics.

6 Conclusion

Modeling the long-term dynamics of complex systems is a challenging problem. Data-driven methods are often constrained by the difficulty in explicitly preserving the inherent spatial interactions of spatiotemporal dynamics in latent vectors. This information loss is exacerbated by new patterns that continuously emerge during the system’s long-term evolution. Therefore, we propose a novel Sparse Diffusion Autoencoder, SparseDiff, a prediction framework that preserves the system’s spatial structure and adapts at test-time. SparseDiff coarsens the full spatial domain state into a probe topology by means of discrete codebook learning to construct a compact skeleton of spatiotemporal dynamics. The pretrained codebook is able to capture the rich dynamic patterns of complex systems. Once training is complete, SparseDiff can adjust the probe topology to adapt to emerging dynamic patterns during testing and without updating weights. On the probe topology, we design an edge-weight-aware graph neural diffusion ordinary differential equation to model spatiotemporal dynamics, thereby predicting the future states of the probes and guiding the diffusion model to efficiently reconstruct back to the full spatial domain. Experiments on simulated and real-world systems show that SparseDiff can achieve optimal long-term prediction and exhibits excellent robustness and generalization ability.

Limitation & Future Work SparseDiff’s spatiotemporal dynamics module relies on the perception of edge weights, therefore the rules for calculating weights will affect its long-term prediction quality. Furthermore, the state initialization of probes is achieved through average aggregation, which may lose high-frequency information at a few grid points. Future research will focus on end-to-end graph structure learning and probe initialization strategies.

7 Author Contributions

Ruikun Li, Huandong Wang and Yong Li launched this research and provided the research outline. Ruikun Li, Jingwen Cheng and Huandong Wang designed the research methods. Jingwen Cheng performed the experiments and prepared the figures & technical appendix. Ruikun Li wrote the original manuscript. Huandong Wang and Yong Li provided critical revisions.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under grant 62171260 and 92270114.

References

- [1] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.
- [2] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.
- [3] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. Vampnets for deep learning of molecular kinetics. *Nature communications*, 9(1):5, 2018.
- [4] Ruikun Li, Huandong Wang, Jinghua Piao, Qingmin Liao, and Yong Li. Predicting long-term dynamics of complex networks via identifying skeleton in hyperbolic space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1655–1666, 2024.
- [5] Ruikun Li, Jingwen Cheng, Huandong Wang, Qingmin Liao, and Yong Li. Predicting the dynamics of complex system via multiscale diffusion autoencoder. *arXiv preprint arXiv:2505.02450*, 2025.

- [6] Pantelis R Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multi-scale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.
- [7] Seungjoon Lee, Mahdi Kooshkbaghi, Konstantinos Spiliotis, Constantinos I Siettos, and Ioannis G Kevrekidis. Coarse-scale pdes from fine-scale observations via machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1), 2020.
- [8] Jingtao Ding, Chang Liu, Yu Zheng, Yunke Zhang, Zihan Yu, Ruikun Li, Hongyi Chen, Jinghua Piao, Huandong Wang, Jiazen Liu, et al. Artificial intelligence for complex network: Potential, methodology and application. *arXiv preprint arXiv:2402.16887*, 2024.
- [9] Katiana Kontolati, Somdatta Goswami, George Em Karniadakis, and Michael D Shields. Learning nonlinear operators in latent spaces for real-time predictions of complex dynamics in physical systems. *Nature Communications*, 15(1):5101, 2024.
- [10] Tao Wu, Xiangyun Gao, Feng An, Xiaotian Sun, Haizhong An, Zhen Su, Shraddha Gupta, Jianxi Gao, and Jürgen Kurths. Predicting multiple observations in complex systems through low-dimensional embeddings. *Nature Communications*, 15(1):2242, 2024.
- [11] Ruikun Li, Huandong Wang, and Yong Li. Learning slow and fast system dynamics via automatic separation of time scales. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4380–4390, 2023.
- [12] Ruikun Li, Huandong Wang, Qingmin Liao, and Yong Li. Predicting the energy landscape of stochastic dynamical system via physics-informed self-supervised learning. *arXiv preprint arXiv:2502.16828*, 2025.
- [13] Marten Lienen, David Lüdke, Jan Hansen-Palmus, and Stephan Günnemann. From zero to turbulence: Generative modeling for 3d flow simulation. *arXiv preprint arXiv:2306.01776*, 2023.
- [14] Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José Miguel Hernández-Lobato, Richard Turner, and Emile Mathieu. On conditional diffusion models for pde simulations. *Advances in Neural Information Processing Systems*, 37:23246–23300, 2024.
- [15] Zeyu Li, Wang Han, Yue Zhang, Qingfei Fu, Jingxuan Li, Lizi Qin, Ruoyu Dong, Hao Sun, Yue Deng, and Lijun Yang. Learning spatiotemporal dynamics with a pretrained generative model. *Nature Machine Intelligence*, 6(12):1566–1579, 2024.
- [16] Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. Diffusionpde: Generative pde-solving under partial observation. *arXiv preprint arXiv:2406.17763*, 2024.
- [17] Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-Francois Chamberland, and Srinivas Shakkottai. Llmzip: Lossless text compression using large language models. *arXiv preprint arXiv:2306.04050*, 2023.
- [18] Harsh Bhatia, Timothy S Carpenter, Helgi I Ingólfsson, Gautham Dharuman, Piyush Karande, Shusen Liu, Tomas Osselstrup, Chris Neale, Felice C Lightstone, Brian Van Essen, et al. Machine-learning-based dynamic-importance sampling for adaptive multiscale simulations. *Nature Machine Intelligence*, 3(5):401–409, 2021.
- [19] Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. In *International Conference on Machine Learning*, pages 17616–17635. PMLR, 2024.
- [20] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.
- [21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

- [22] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- [23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [24] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.
- [25] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.
- [26] B. P. Chamberlain, J. Rowbottom, M. Gorinova, S. Webb, E. Rossi, and M. M. Bronstein. Grand: Graph neural diffusion. 2021.
- [27] Mark Veillette, Siddharth Samsi, and Chris Mattioli. Sevir: A storm event imagery dataset for deep learning applications in radar and satellite meteorology. *Advances in Neural Information Processing Systems*, 33:22009–22019, 2020.
- [28] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*.
- [29] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.
- [30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.
- [31] Han Gao, Sebastian Kaltenbach, and Petros Koumoutsakos. Generative learning for forecasting the dynamics of high-dimensional complex systems. *Nature Communications*, 15(1):8904, 2024.
- [32] Zhibing Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint*, 2020.
- [33] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.
- [34] Qi Wang, Pu Ren, Hao Zhou, Xin-Yang Liu, Zhiwen Deng, Yi Zhang, Ruizhi Chengze, Hongsheng Liu, Zidong Wang, Jian-Xun Wang, et al. P2c2net: Pde-preserved coarse correction network for efficient prediction of spatiotemporal dynamics. *arXiv preprint arXiv:2411.00040*, 2024.
- [35] Alberto Solera-Rico, Carlos Sanmiguel Vila, Miguel Gómez-López, Yuning Wang, Abdulrahman Almashjary, Scott TM Dawson, and Ricardo Vinuesa. β -variational autoencoders and transformers for reduced-order modelling of fluid flows. *Nature Communications*, 15(1):1361, 2024.
- [36] Felix P Kemeth, Tom Bertalan, Thomas Thiem, Felix Dietrich, Sung Joon Moon, Carlo R Laing, and Ioannis G Kevrekidis. Learning emergent partial differential equations in a learned emergent space. *Nature communications*, 13(1):3318, 2022.
- [37] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

- [38] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024.
- [39] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1921–1930, 2023.
- [40] Xinyao Fan, Yueying Wu, Chang Xu, Yuhao Huang, Weiqing Liu, and Jiang Bian. Mg-tsd: Multi-granularity time series diffusion models with guided learning process. *arXiv preprint arXiv:2403.05751*, 2024.
- [41] Lifeng Shen, Weiyu Chen, and James Kwok. Multi-resolution diffusion models for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.
- [42] Dule Shu, Zijie Li, and Amir Barati Farimani. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 2023.
- [43] Jan-Hendrik Bastek, WaiChing Sun, and Dennis M Kochmann. Physics-informed diffusion models. *arXiv preprint arXiv:2403.14404*, 2024.
- [44] Tiansi Li, Luca Biferale, Fabio Bonacorso, Martino Andrea Scarpolini, and Michele Buzzicotti. Synthetic lagrangian turbulence by generative diffusion models. *Nature Machine Intelligence*, 6(4):393–403, 2024.
- [45] Salva Röhling Cachay, Bo Zhao, Hailey Joren, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *Advances in neural information processing systems*, 36:45259–45287, 2023.
- [46] Xingzhuo Guo, Yu Zhang, Baixu Chen, Haoran Xu, Jianmin Wang, and Mingsheng Long. Dynamical diffusion: Learning temporal dynamics with diffusion models. *arXiv preprint arXiv:2503.00951*, 2025.
- [47] Qian Zhao, David B. Lindell, and Gordon Wetzstein. Learning to solve PDE-constrained inverse problems with graph networks. *arXiv preprint*, 2022.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

A Data Generation

Below, we provide an overview of the dynamics and data generation process for each complex system. **LambdaOmega (LO) system** is governed by

$$\begin{cases} \dot{u}_t = \mu_u \Delta u + (1 - u^2 - v^2)u + \beta(u^2 + v^2)v \\ \dot{v}_t = \mu_v \Delta v + (1 - u^2 - v^2)v - \beta(u^2 + v^2)u, \end{cases} \quad (5)$$

where Δ is the Laplacian operator. Here, we set μ_u and μ_v to 0.5, while β is 1.0.

Navier-Stokes (NS) system is governed by:

$$\dot{\omega}_t + (\mathbf{u} \cdot \nabla) \omega = \nu \Delta \omega + f, \quad (6)$$

$$\nabla^2 \psi = -\omega, \quad (7)$$

$$\mathbf{u} = (u, v) = \left(\frac{\partial \psi}{\partial y}, -\frac{\partial \psi}{\partial x} \right), \quad (8)$$

$$\omega = (\nabla \times \mathbf{u}) \cdot \hat{\mathbf{z}} = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}, \quad (9)$$

where $f = A(\sin(2\pi(x + y + s)) + \cos(2\pi(x + y + s)))$ is the driving force, and ν is the viscosity coefficient. Parameter values used in this simulation are set as follows: $\nu = 1.0$, forcing amplitude $A = 0.1$, and phase shift $s = 0$.

Swift-Hohenberg (SH) system is simulated as:

$$\frac{\partial u}{\partial t} = ru - 2\Delta u - \Delta^2 u + gu^2 - u^3, \quad (10)$$

where r is the linear instability parameter, g controls the strength of the quadratic nonlinearity, and Δ^2 denotes the biharmonic operator. In this system we set r to 0.7 and g 1.0 separately.

Cylinder flow (CY) system is governed by:

$$\begin{cases} \dot{u}_t = -u \cdot \nabla u - \frac{1}{\alpha} \nabla p + \frac{\beta}{\alpha} \Delta u, \\ \dot{v}_t = -v \cdot \nabla v + \frac{1}{\alpha} \nabla p - \frac{\beta}{\alpha} \Delta v. \end{cases} \quad (11)$$

where α is set to 1.0, and β corresponds to the dynamic viscosity μ as defined in Equation 12.

The systems under study are simulated across a diverse range of initial conditions. Temporal down-sampling by a factor of 10 and 25 are applied to the LO and NS systems separately to reduce redundancy, while all the systems are spatially rescaled to a uniform 128×128 resolution.

The Cylinder flow system is modeled via the lattice Boltzmann method (LBM) [6], capturing complex vortex shedding phenomena governed by the Navier-Stokes equations in the presence of a cylindrical obstacle. The simulation operates on a lattice velocity grid, where relaxation dynamics are dictated by the kinematic viscosity and the Reynolds number. To ensure data quality, we begin recording after the flow reaches a statistically steady turbulent regime. The outputs are resampled in time by a factor of 300 and spatially interpolated to a 128×64 grid. We generate 50 training and 20 testing trajectories across varying flow conditions, with Reynolds numbers uniformly sampled: 10 training samples in the range $\text{Re} \in [100, 500]$ and 10 out-of-distribution (OOD) samples in $\text{Re} \in [500, 1000]$. Based on the relation

$$\mu = \frac{\rho U_m D}{\text{Re}} \quad (12)$$

where $\rho = 1$, $U_m = 0.08$, and $D = 0.2$, the corresponding dynamic viscosity spans $\mu \in [3.2 \times 10^{-5}, 1.6 \times 10^{-4}]$ for training and $\mu \in [1.6 \times 10^{-5}, 3.2 \times 10^{-5}]$ for OOD cases.

To standardize input scales and facilitate stable training, we apply min-max normalization independently across all channels.

B Model Architecture

Our model is composed of three key modules: a Codebook-based Sparse Encoder for spatiotemporal discretization, a Probe-graph Diffusive Predictor for latent dynamics modeling, and a Guided Diffusion Decoder for reconstructing full-resolution system states. Below, we detail the architectural settings used in each component.

```
# --- Sparse Encoder ---
hidden_dim = 1024      # hidden dimension of MLP
embedding_dim = 512      # Latent dimension (d)
num_embeddings = M      # Codebook size, hyperparameter

# --- Diffusive Predictor ---
input_steps = 10      # number of lookback steps
feature_dim = 256      # Feature dimension
num_heads = 8      # Attention heads for calculating Matrix A
ODE_method = 'rk4'      # Numerical solver for ODE integration

# --- Unconditioned Diffusion ---
n_channels = 128      # Base number of channels
ch_mults = [1, 2, 2]      # Channel multiplier for each resolution level
is_attn = [False, False, True]      # Whether to apply self-attention
dropout = 0.1      # Dropout rate in residual blocks
n_blocks = 2      # Number of residual blocks per resolution level
```

Table 3: Trainable parameter counts (in millions).

Model	SparseDiff			FNO	ConvLSTM	UNet	G-LED
	Encoder	Predictor	Diffusion				
Params (M)	2.3×10^{-2}	1.32	25.8	23.90	5.47	10.91	26.3

C Baseline Implementation

We provide a brief description of the baseline methods used for comparison in our experiments. These methods represent a diverse set of state-of-the-art approaches for modeling spatiotemporal dynamics. Their corresponding trainable parameter counts are summarized in Table 3.

- **FNO** [28]: Fourier Neural Operator leverages fast Fourier transforms to model spatially continuous operators, enabling efficient learning of solution mappings for partial differential equations. It is widely adopted for learning complex physical dynamics.
- **ConvLSTM** [29]: ConvLSTM integrates convolutional structures into recurrent networks, allowing spatial correlations to be preserved while capturing temporal dependencies. It is a standard baseline for video and sequence-based spatial forecasting tasks.
- **UNet** [30]: UNet employs an encoder-decoder structure with skip connections to effectively combine global context and local details. It is particularly suitable for dense prediction tasks involving structured outputs.
- **G-LED** [31]: G-LED is a generative latent evolution model that models temporal dynamics using autoregressive attention in latent space, and reconstructs full-resolution outputs using a Bayesian diffusion model. It achieves strong performance on high-dimensional physical systems.

D Additional Results

D.1 Application to Irregular or Sparse Spatial Domain

While our current implementation operates on 2D regular grids, SparseDiff can be applied to irregular or sparse spatial data by first transforming the inputs into a regular grid through interpolation or zero-filling, enabling processing within the same framework.

To validate this, we conduct two experiments:

a. Sparse sampling on Navier-Stokes system: On the Navier-Stokes system, we randomly sample only 10% of grid points. These sparse samples are then interpolated onto a 128×128 regular grid before being fed into SparseDiff. Despite the sparsity, SparseDiff outperforms baselines that are given full-resolution inputs, as shown in Table 4.

Table 4: Sparse sampling on the Navier-Stokes system. We report average metrics of 100 prediction steps.

Method	RMSE $\times 10^{-2} \downarrow$	SSIM $\times 10^{-1} \uparrow$
FNO (full grid)	15.99	7.26
G-LED (full grid)	12.33	8.10
SparseDiff (full grid)	11.39	8.36
SparseDiff (10% observations)	12.27	8.19

b. Irregular 2D wave equation dataset [47]: We also evaluate SparseDiff on the 2D wave equation with observations sampled on irregular meshes. These inputs are first completed onto regular grids using interpolation within observed regions and zero-filling in excluded regions before being processed by the model. Compared to FNO, SparseDiff maintains stable prediction accuracy (Table 5).

Table 5: Results on the irregular 2D wave equation dataset [47] of 100 prediction steps.

Method	RMSE $\times 10^{-2} \downarrow$	SSIM $\times 10^{-1} \uparrow$
FNO	23.76	7.81
SparseDiff	15.58	8.25

While these results demonstrate practical applicability to irregular or sparse inputs, we acknowledge certain limitations of this approach:

- For highly non-uniform spatial distributions, interpolating to a uniform grid can be problematic. Also, the sparse observation may be too under-sampled to allow reliable interpolation, leading to unnecessary computational cost or significant reconstruction errors.
- If the distribution of spatial inputs shifts between training and testing, performance may degrade, as our current model lacks coordinate-querying capabilities (unlike operator-learning models such as neural operators), and cannot generalize across arbitrary spatial layouts.

Nevertheless, the core design of SparseDiff is not inherently tied to regular grids. The key innovation of SparseDiff lies in its latent probe representation, which aggregates and models regional dynamics through sparse, learnable units. This representation is inherently flexible and does not require regular spatial structures.

The main constraint stems from the UNet-based diffusion decoder, which requires regular-grid input. However, this is a design choice rather than a fundamental limitation of the method. In future work, the diffusion decoder could be replaced with architectures that naturally support irregular domains, such as Graph Neural Networks. Alternatively, it could be substituted with other super-resolution models, including transformer-based approaches and operator-learning models, to further extend SparseDiff to irregular spatial settings.

D.2 Visualization

We provide long-term prediction visualizations on three representative systems: NS, SH, and CY. For each case, the left column shows the ground truth, the middle column presents the prediction reconstructed via the diffusion decoder, and the right column shows the vanilla reconstruction by directly filling each codeword region with its corresponding probe value.

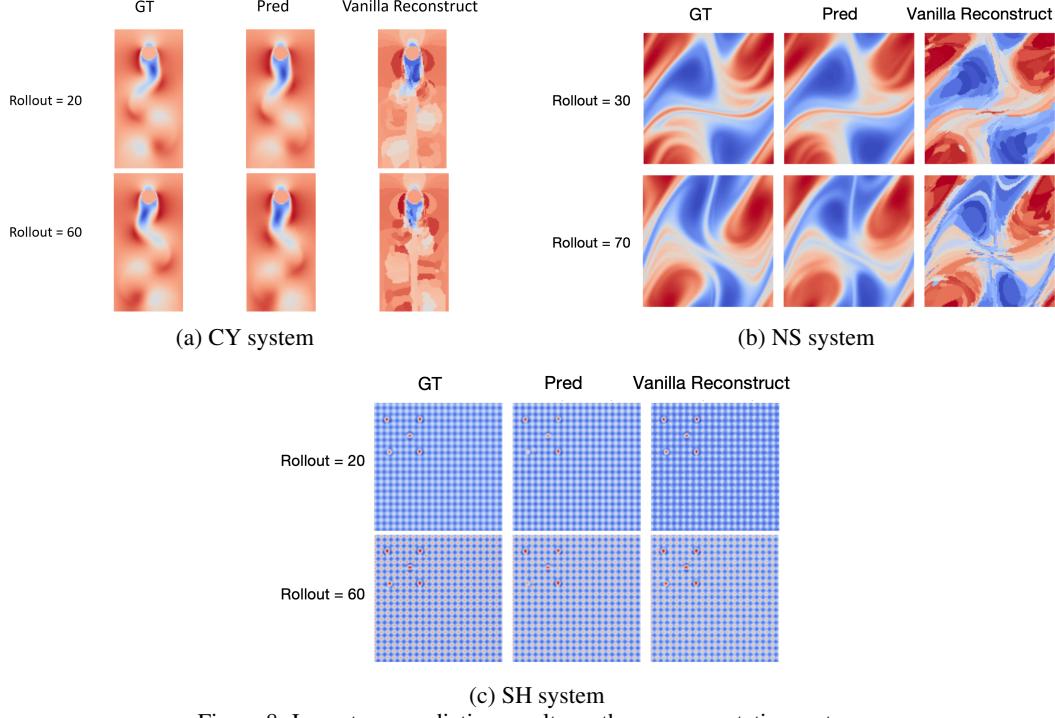


Figure 8: Long-term prediction results on three representative systems.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Can be found at the last paragraph of the introduction section.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Can be found at the limitation paragraph of the conclusion section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Can be found in the data generation and supplementary materials sections of the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: The attached file submitted contains the code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Can be found in the data generation section of the appendix and submitted code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: See each table caption in the experiment section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: Can be found in the time cost section in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: The authors have read and conform the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: Can be found in the impact statement section in the appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have cited the original paper that produced the code package or dataset.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New Assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and Research with Human Subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.