Efficient Generative Transformer Operators for Million-Point PDEs

Armand Kassaï Koupaï[1]* **Lise Le Boudec**[1]* **Patrick Gallinari**[1,2]
[1] Sorbonne Université, CNRS, ISIR, 75005 Paris, France
[2] Criteo AI Lab, Paris, France

Abstract

We introduce **ECHO**, a transformer-based neural operator for large-scale PDE modeling that learns to *generate full trajectories in compressed spatio-temporal latent spaces*. Scaling neural operators to high-resolution systems remains challenging: dense grids are computationally prohibitive, and autoregressive solvers accumulate errors over long horizons. ECHO addresses both issues by jointly compressing space and time and replacing step-by-step prediction with trajectory-level generation. ECHO combines a hierarchical encoder–decoder achieving up to 100× compression, a staged training strategy for high-resolution fidelity, and a latent generative process that models distributions over complete trajectories, improving long-term consistency. This formulation enables a single model to handle forward prediction, backward prediction, and interpolation. Across large-scale 2D and 3D benchmarks, ECHO achieves state-of-the-art performance and scales to million-point simulations with complex dynamics.

1 Introduction

Neural networks have emerged as a promising alternative to classical numerical solvers for modeling physical dynamics and solving partial differential equations (PDEs) (Long et al., 2018; Raissi et al., 2019; de Bézenac et al., 2019). Neural operators (NOs) extend these approaches by learning mesh-independent mappings between function spaces, enabling generalization across domains, resolutions, and discretizations (Lu et al., 2021; Li et al., 2021, 2023b; Serrano et al., 2023), and supporting parametric (Cohen & Devore, 2015; Koupaï et al., 2024) and multi-physics problems (McCabe et al., 2024; Herde et al., 2024). Despite this progress, scaling NOs to high-resolution, large-scale systems remains a fundamental challenge. Interactions over millions of points quickly exhaust computational resources, while autoregressive time-stepping accumulates errors over long horizons (Lippe et al., 2023; Pedersen et al., 2025). As a result, existing approaches either focus on simplified settings, such as regular grids (Holzschuh et al., 2025; Rozet et al., 2025), or limit problem complexity when addressing irregular geometries and high-frequency dynamics. Latent-space methods alleviate some of these costs (Alkin et al., 2024; Serrano et al., 2024), but still struggle to jointly compress space and time without degrading reconstruction quality, especially at scale (Koupaï et al., 2025; Wen et al., 2025; Elrefaie et al., 2024).

This motivates a central question: *What is the correct paradigm to scale surrogate models accurately and robustly to large-scale, arbitrary domains?* We explore a formulation that couples strong spatio-temporal compression with trajectory-level generation, learning to represent dynamics in compact latent spaces while preserving relevant physical information. This enables models to generate complete trajectory segments, enforcing global temporal coherence, reducing error accumulation, and improving scalability to large-scale systems. We operationalize this idea through three complementary principles, illustrated in fig. 1.

*Equal contribution. Correspondence: armand.kassai[at]isir.upmc.fr, lise.leboudec[at]isir.upmc.fr

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

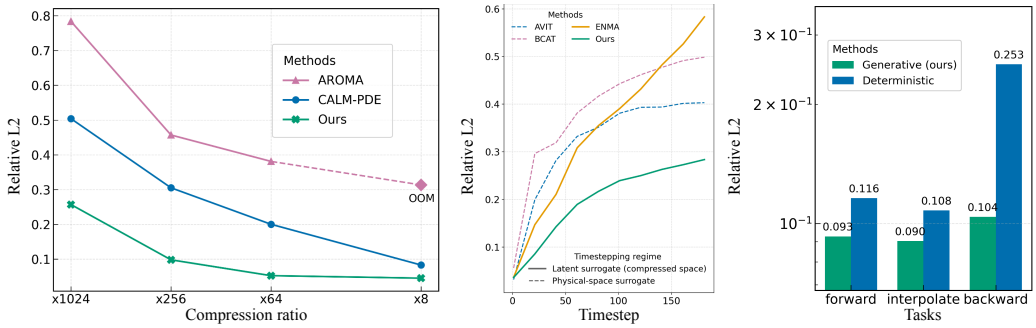


Figure 1: **Experimental analysis of ECHO.** Left: reconstruction error versus compression ratio on *Vorticity*. Hierarchical, iterative compression improves accuracy, especially at high compression ratios. Middle: long-range rollouts on *Gray-Scott*. Given the first 3 frames, each model predicts the next 197 frames; frames beyond $t = 40$ lie outside the training horizon. Errors are averaged over test trajectories. Full-trajectory generation accumulates less error than latent autoregressive and physical-space baselines. Right: generative ECHO versus its deterministic variant on forward prediction, interpolation and backward prediction on *Rayleigh-Bénard*. All panels report the relative L^2 error (lower is better).

(i) Progressive hierarchical spatio-temporal compression. Most encoders downsample only spatial dimensions and perform compression in a single step, leading to poor scalability and information loss (Mousavi et al., 2025; Alkin et al., 2024; Serrano et al., 2024; Wang & Wang, 2024). We introduce a hierarchical encoder–decoder that progressively compresses both space and time, yielding compact yet faithful spatio-temporal representations (fig. 1 – *Left*). **(ii) Trajectory-level generation.** Autoregressive prediction remains the dominant paradigm (McCabe et al., 2024; Hao et al., 2024; Wang et al., 2025b; Liu et al., 2025), but suffers from error drift. We instead generate complete trajectory segments, enforcing horizon-wide consistency, mitigating error drift, and improving long-term stability (fig. 1 – *Center*). **(iii) Generative modeling in compressed latent space.** We adopt a stochastic formulation (Zhou et al., 2025a) that models distributions over full trajectories rather than deterministic predictions (Huang et al., 2024; Serrano et al., 2025; Koupaï et al., 2025), enabling robustness to partial observations and uncertainty (fig. 1 – *Right*).

We instantiate these principles in **ECHO**, a transformer-based neural operator for large-scale PDE modeling. By combining hierarchical compression with trajectory-level generative modeling, ECHO enables efficient learning and inference on million-point systems defined on arbitrary domains, where most SOTA neural operators fail. Critical to this capability is the efficient spatio-temporal representation. Figure 1 demonstrates the impact of each component: (left) significantly improved accuracy at high compression ratios as an enabler of large-scale applications, (center) a trajectory generation mechanism that reduces long-horizon error accumulation, and (right) a generative modeling paradigm that offers consistent gains over deterministic baselines. ECHO achieves state-of-the-art performance on large-scale 2D and 3D PDE benchmarks, scaling to simulations with up to 20 million points, where existing methods fail.

In addition, we introduce a staged training strategy that enables high-fidelity learning under memory constraints and a unified formulation that supports forward prediction, backward prediction, interpolation, and long-horizon forecasting within a single model. This enables us to address a variety of problems in a zero-shot setting.

2 Problem Setting

We consider time-dependent partial differential equations (PDEs) defined on a spatial domain $\Omega \subset \mathbb{R}^d$ over a time interval $[0, T]$. Each instance is specified by an initial condition $\boldsymbol{u}^0 \in L^2(\Omega, \mathbb{R}^{d_u})$ and a set of parameters $\gamma = (\boldsymbol{b}, \boldsymbol{f}, \boldsymbol{c})$, which include boundary conditions $\boldsymbol{b} \in L^2(\partial\Omega \times [0, T], \mathbb{R}^{d_b})$, a

forcing term $\boldsymbol{f} \in L^2(\Omega \times [0,T], \mathbb{R}^{d_f})$, and PDE coefficients $\boldsymbol{c}$. The governing system is

$$\mathcal{N}[\boldsymbol{u}; \boldsymbol{c}, \boldsymbol{f}](x,t) = 0, \qquad (x,t) \in \Omega \times (0,T], \tag{1}$$

$$\mathcal{B}[\boldsymbol{u}; \boldsymbol{b}](x,t) = 0, \qquad (x,t) \in \partial\Omega \times [0,T], \tag{2}$$

$$\boldsymbol{u}(x,0) = \boldsymbol{u}^0(x), \qquad x \in \Omega, \tag{3}$$

where $\mathcal{N}$ is a (possibly nonlinear) differential operator, $\mathcal{B}$ encodes the boundary conditions, and $\boldsymbol{u} : [0,T] \times \Omega \to \mathbb{R}^{d_u}$ denotes the solution field.

In contrast to classical numerical solvers, fully data-driven neural solvers are not provided with an explicit PDE and must therefore be conditioned either on explicit PDE parameters γ or on sequences of observed states. For training, we assume access to a finite dataset $\mathcal{D}\texttt{tr}$ of N trajectories, observed on a free-form spatial grid $\mathcal{X}\texttt{tr}$ at discrete times $\mathcal{T} \subset [0,T]$. Each trajectory consists of $|\mathcal{X}|$ mesh points, with $|\mathcal{X}|$ potentially very large in real-world problems. At test time, trajectories are observed on a spatial grid $\mathcal{X}\texttt{te}$, which may differ from $\mathcal{X}\texttt{tr}$.

To our knowledge, **ECHO** is the first generative operator to address forward prediction, backward prediction and interpolation under a unified formalism while operating in a compressed spatio-temporal latent space on arbitrary domains, enabling scalable modeling of high-resolution PDEs. Details of the inference setup, architecture, and training strategy are provided in the following sections.

2.1 Inference Model

Since directly modeling PDE dynamics in physical space becomes computationally prohibitive at large scale, ECHO adopts an *encode–generate–decode* paradigm (see fig. 2). This formulation allows us to handle partial observations on regular or irregular meshes at arbitrary resolutions.

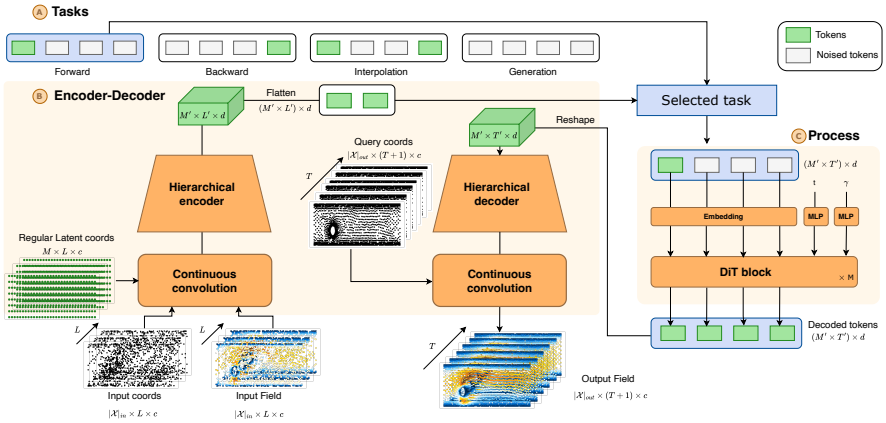


Figure 2: Architecture of the ECHO framework. ECHO comprises two components: (B) a convolutional auto-encoder and (C) a DiT-based generative process. The auto-encoder uses continuous convolutions to ingest irregular input grids of arbitrary size, maps the dynamics to a regular latent grid, and hierarchically compresses it; the decoder mirrors this hierarchy and applies a final continuous convolution, enabling queries at arbitrary output locations. The DiT module is trained with a flow-matching objective to denoise latent tokens, optionally conditioned on PDE parameters. This design allows ECHO to handle irregular grids and support multiple inference tasks (A).

Inference setting. At inference time, the objective is to reconstruct a full spatio-temporal trajectory from a limited set of observations. Let $\boldsymbol{u}^{0:T} \in \mathbb{R}^{|\mathcal{X}| \times (T+1) \times c}$ denote the complete trajectory defined over $|\mathcal{X}|$ spatial locations and $T+1$ time steps with c physical channels. We observe only L states at times $\{t_0, \ldots, t_{L-1}\} \subset [0,T]$ (see fig. 2 - A). We denote the observed and missing subsets as

$$\mathcal{O} = \{\boldsymbol{u}^{t_\ell} : t_\ell \in \{t_0, \ldots, t_{L-1}\}\}, \quad \mathcal{M} = \{\boldsymbol{u}^t : t \notin \{t_0, \ldots, t_{L-1}\}\},$$

with $\mathcal{O} \cap \mathcal{M} = \emptyset$. Using a binary mask $m \in \{0,1\}^{T+1}$, we write

$$\boldsymbol{u}^{\mathcal{O}} = \boldsymbol{u}^{0:T} \odot m, \quad \boldsymbol{u}^{\mathcal{M}} = \boldsymbol{u}^{0:T} \odot \bar{m}, \quad \boldsymbol{u}^{\mathcal{O}} + \boldsymbol{u}^{\mathcal{M}} = \boldsymbol{u}^{0:T}.$$

Latent formulation. Rather than operating in physical space, inference is performed in a compressed spatio-temporal latent space. Observations are encoded by a hierarchical encoder E_ϕ into latent tokens $\boldsymbol{z}^{\mathcal{O}'} = E_\phi(\boldsymbol{u}^{\mathcal{O}}) \in \mathbb{R}^{M' \times L' \times d}$, defined on a regular latent grid. Here, M' and L' denote the number of spatial and temporal tokens, and d is the latent dimension (see fig. 2 - B). The temporal horizon is also compressed, with T' corresponding to the latent counterpart of T. The inference problem is thus recast as *latent trajectory completion*: given observed tokens $\boldsymbol{z}^{\mathcal{O}'}$, we aim to predict the missing tokens $\boldsymbol{z}^{\mathcal{M}'}$. At the decoding stage, the latter will be mapped back to the physical space to get the reconstruction of $\boldsymbol{u}^{\mathcal{M}}$.

Latent trajectory generation. The generative model $\mathcal{G}_\theta$ predicts the full latent trajectory $\boldsymbol{z}^{0:T'}$ using a transformer trained with a flow-matching objective, made tractable thanks to the reduced number of tokens. During generation, missing tokens $\boldsymbol{z}^{\mathcal{M}'}$ are initialized from Gaussian noise, while observed tokens $\boldsymbol{z}^{\mathcal{O}'}$ and (optionally) PDE parameters $\boldsymbol{\gamma}$ provide conditioning (see fig. 2 - C).

Trajectory generation is performed by solving the ODE

$$\boldsymbol{z}^{0:T'} = \texttt{ODESOLVE}\Big(\mathcal{G}_\theta(\boldsymbol{z}_r^{\mathcal{M}'}, \boldsymbol{z}^{\mathcal{O}'}, \boldsymbol{\gamma}, r)\Big), \quad \boldsymbol{z}_0^{\mathcal{M}'} \sim \mathcal{N}(0, I), \tag{4}$$

where $r \in [0, 1]$ is the denoising index. We use the midpoint method as the ODE solver. Because the entire temporal sequence is resolved simultaneously, temporal drift is explicitly decoupled from the sequence length. The generation error is bounded by the flow-matching transport error and the autoencoder's information bottleneck, allowing ECHO to maintain stability over significantly longer physical horizons than iterative methods.

Decoding. The predicted latent trajectory $\hat{\boldsymbol{z}}^{0:T'}$ is finally mapped back to the physical space using a decoder D_ψ, yielding the reconstructed trajectory $\hat{\boldsymbol{u}}^{0:T}$.

Example: forward forecasting. ECHO supports multiple inference tasks by varying the observed set $\mathcal{O}$. For forward forecasting, $\mathcal{O}$ contains the first L frames. These are encoded as $\boldsymbol{z}^{0:L'-1} = E_\phi(\boldsymbol{u}^{0:L-1})$, and the model predicts the remaining tokens $\{\boldsymbol{z}^{L'}, \ldots, \boldsymbol{z}^{T'}\}$ to complete the trajectory.

The full pipeline can be summarized as:

$$\boldsymbol{u}^{\mathcal{O}} \in \mathbb{R}^{|\mathcal{X}| \times L \times c} \xrightarrow{\text{encode}} \boldsymbol{z}^{\mathcal{O}'} \in \mathbb{R}^{M' \times L' \times d} \xrightarrow{\text{mask}} \boldsymbol{z}^{\mathcal{O}' \cup \mathcal{M}'} \in \mathbb{R}^{M' \times T' \times d} \xrightarrow{\text{generate}} \hat{\boldsymbol{z}}^{0:T'} \in \mathbb{R}^{M' \times T' \times d} \xrightarrow{\text{decode}} \hat{\boldsymbol{u}}^{0:T} \in \mathbb{R}^{|\mathcal{X}| \times (T+1) \times c}$$

Extensions. By varying $\mathcal{O}$, the same framework supports interpolation, backward prediction, and conditional or unconditional generation (see Appendix D.1). For long-horizon prediction, we further apply segment-wise autoregressive generation. As in neural operator frameworks, outputs can be queried at arbitrary spatial locations, enabling tasks such as super-resolution and inpainting.

3 Method

3.1 Architecture

Encoding. A core component of ECHO is a *hierarchical spatio-temporal encoder* that progressively compresses inputs. High-quality reconstruction at strong compression ratios is difficult to achieve with a single-step projection to a small latent space. Instead, we adopt a sequential compression strategy, where information is gradually aggregated across space and time, allowing early layers to preserve fine-scale structure before further compression.

We first embed irregular spatio-temporal data $\boldsymbol{u}^{\mathcal{O}}$ onto a regular latent grid, and then apply convolutional stages that reduce spatial and temporal resolution in a structured, multi-scale manner. This enables high compression while preserving fine-scale dynamics, which is critical for stable generation at scale. Encoding proceeds in two steps:

- **Latent grid mapping.** We map irregular spatio-temporal inputs onto a fixed regular grid $\boldsymbol{\Xi} = \{\xi_j\}_{j=1}^{S}$ using a continuous convolution (Hagnberger et al., 2025). Let $f : \mathcal{X} \to \mathbb{R}^{C_{\text{in}}}$ denote the input field defined at positions $\{x_p\}_{p=1}^{N}$, and k a learned kernel. The output at latent query ξ_j and

channel o is:

$$(f * k)_o(\xi_j) = \sum_{i=1}^{C_{\text{in}}} \sum_{p \in \text{RF}(\xi_j)} f_i(x_p)\, k_{i,o}(\xi_j - x_p). \tag{5}$$

Here, $\text{RF}(\xi_j)$ denotes the set of input points contributing to ξ_j, and the kernel $k_{i,o}$ is parameterized by an MLP over relative displacements $\xi_j - x_p$. This continuous-convolution layer is the only operation that directly couples the irregular physical mesh to the latent grid.

A naive implementation evaluates all pairs in $\mathcal{X} \times \Xi$, requiring $\mathcal{O}(NS)$ operations. To ensure scalability, we restrict each query ξ_j to a local neighborhood $\mathcal{N}(\xi_j)$ of size $K \ll N$, yielding

$$(f * k)_o(\xi_j) = \sum_{i=1}^{C_{\text{in}}} \sum_{p \in \mathcal{N}(\xi_j)} f_i(x_p)\, k_{i,o}(\xi_j - x_p). \tag{6}$$

The mapping $j \mapsto \mathcal{N}(\xi_j)$ is computed using a spatial index, reducing complexity to approximately $\mathcal{O}(S \log N + SK)$, and is cached when meshes are reused.

We further reduce memory by factorizing the last layer of the kernel MLP: hidden features are first aggregated over the K neighbors, followed by channel mixing. This avoids materializing tensors of size $S \times K \times C_{\text{in}} \times C_{\text{out}}$, reducing memory by a factor $\frac{KC_{\text{out}}}{H+1}$, per latent query where H is the hidden width.

- **Spatio-temporal compression.** Once the irregular field is mapped to $\boldsymbol{u}^{\mathcal{O}}(\Xi)$, the remainder of the encoder operates on a regular grid. We can therefore use standard spatial and temporal convolutions. In this way, ECHO incurs the irregular-mesh cost only at the input and output layers. We stack `compress` blocks, which reduce spatial or temporal resolution, and `residual` blocks, which preserve resolution while enriching the representation. All temporal convolutions are causal, so a state at time j depends only on states at times $i \leq j$. This hierarchical design progressively reduces the number of spatio-temporal tokens before the transformer stage. As a result, the generative model operates on compact latent trajectories rather than on the original physical mesh, which is essential for efficient large-scale trajectory generation.

Decoding. The decoder mirrors the encoder. Transposed convolutions reverse the spatio-temporal compression, and a final continuous convolution interpolates latent tokens to arbitrary physical query locations. This design enables querying at arbitrary spatial locations and supports tasks such as super-resolution and inpainting (see section D.1).

Transformer Process The ECHO process component is a transformer architecture trained with a flow-matching loss. It builds on the MM-DiT block (Esser et al., 2024). Given observed latent tokens $\boldsymbol{z}^{\mathcal{O}'}$, unobserved $\boldsymbol{z}^{\mathcal{M}'}$ are initialized with a noise distribution and the model is trained to reconstruct the full trajectory $\boldsymbol{z}^{0:T'}$ in the compact latent space. Thanks to the high compression ratio, global attention remains tractable—unlike recent PDE transformers that resort to neighborhood attention on raw fields (Holzschuh et al., 2025). A key feature of ECHO is its latent conditioning strategy, which is designed to support generative modeling of PDEs defined on arbitrary geometries. Instead of conditioning via adaptive normalization alone, as usually done, we concatenate the observed regularized latent tokens with noise:

$$\texttt{concat}(\boldsymbol{z}^{\mathcal{O}'}, \boldsymbol{z}^{\mathcal{M}'}_{\texttt{noise}}), \quad \text{where} \quad \mathcal{O}' \cup \mathcal{M}' = [0, T'].$$

This provides an in-context mechanism where observed frames guide noised ones. Denoising index r is injected explicitly; it is embedded and added to normalization layers via AdaLN. The model thus processes the entire spatio-temporal trajectory as a unified structure. When PDE parameters $\boldsymbol{\gamma}$ are available, ECHO conditions on both $\boldsymbol{\gamma}$ and the observed tokens $\boldsymbol{z}^{\mathcal{O}'}$. Parameters are encoded by a lightweight MLP and injected into each transformer block via AdaLN (Zhou et al., 2025a; Li et al., 2025b; Zhou et al., 2025b).

3.2 Training

Training generative models on full high-resolution trajectories is computationally and memory intensive, especially on irregular meshes, and existing methods often fail to scale to large problems. ECHO addresses this with a three-stage strategy that reduces memory requirements while preserving

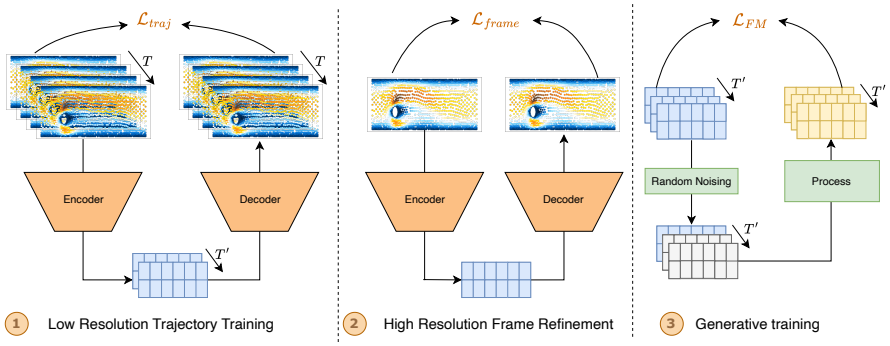


Figure 3: Three-stage training strategy for million-point trajectory generation. ECHO's auto-encoder is first trained for low-resolution trajectory reconstruction, then refined on high-resolution single frames. Both stages use reconstruction objectives. The generative processor is then trained separately in the frozen latent space using flow matching.

high-fidelity reconstruction. First, the auto-encoder learns spatio-temporal compression on low-resolution trajectory segments. Second, it is refined for spatial compression on high-resolution individual frames. Finally, the transformer processor is trained in the frozen latent space with a flow-matching objective.

Stage 1: Spatio-temporal compression on subsampled trajectories. Training on dense, long trajectories is prohibitive. We therefore split trajectories into sub-trajectories using a sliding window and spatially subsample each frame by a factor between 0.2 and 0.5. This avoids out-of-memory errors while exposing the model to temporal contexts of 10–50 frames. The auto-encoder is trained with a relative L^2 reconstruction loss. Let B denote a batch of N_B trajectories:

$$\mathcal{L}_{\text{traj}} = \frac{1}{N_B} \sum_{i=1}^{N_B} \frac{\|\hat{\boldsymbol{u}}_i^{\text{trajectory}} - \boldsymbol{u}_i^{\text{trajectory}}\|}{\|\boldsymbol{u}_i^{\text{trajectory}}\|},$$

Stage 2: Spatial refinement on high-resolution frames. Although Stage 1 provides strong spatio-temporal compression, it does not fully capture high-resolution information. Since high-resolution trajectory-wise training exceeds GPU memory, we refine the auto-encoder frame-wise on high-resolution frames, spatially subsampled by a factor between 0.5 and 1.0. This shorter refinement stage incorporates fine-scale spatial information at substantially lower cost. Let B be a batch of N_B trajectories, the analogous frame-wise loss is

$$\mathcal{L}_{\text{frame}} = \frac{1}{N_B} \sum_{i=1}^{N_B} \frac{\|\hat{\boldsymbol{u}}_i^{\text{frame}} - \boldsymbol{u}_i^{\text{frame}}\|}{\|\boldsymbol{u}_i^{\text{frame}}\|},$$

Stage 3: Flow-matching in latent space. The auto-encoder is then frozen, and the transformer processor is trained in latent space using the flow-matching objective described in section 3.1. Given an encoded latent trajectory $\boldsymbol{z}$, we randomly noise 80% of its latent frames, denoted $\boldsymbol{z}^{\mathcal{M}'}$, and condition on the remaining observed 20%, denoted $\boldsymbol{z}^{\mathcal{O}'}$; see fig. 3. The model learns the conditional distribution $p(\boldsymbol{z}^{\mathcal{M}'} \mid \boldsymbol{z}^{\mathcal{O}'}, \boldsymbol{\gamma})$ through a flow-based transport from the base distribution $p_0 = \mathcal{N}(0, I)$ to the target distribution. The transport is parameterized by the ODE

$$\frac{d\mathbf{z}^r}{dr} = v(\mathbf{z}^r, r), \tag{7}$$

where $r \in [0, 1]$ is a denoising index and v is a neural-network velocity field. During training, intermediate points are sampled as

$$\mathbf{z}^r = r\boldsymbol{z}^{\mathcal{M}'} + (1 - r)\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I). \tag{8}$$

Conditioned on $\boldsymbol{z}^{\mathcal{O}'}$ and, when available, PDE parameters $\boldsymbol{\gamma}$, the transformer takes $(\mathbf{z}^r, \boldsymbol{z}^{\mathcal{O}'}, \boldsymbol{\gamma}, r)$ as input and predicts the transport direction. It is trained with the following loss:

$$\mathcal{L}_{\texttt{FM}} = \mathbb{E}_{\boldsymbol{\epsilon}, r}\left[\left\|\mathcal{G}_\theta(\mathbf{z}^r, \boldsymbol{z}^{\mathcal{O}'}, \boldsymbol{\gamma}, r) - (\boldsymbol{z}^{\mathcal{M}'} - \boldsymbol{\epsilon})\right\|_2^2\right], \tag{9}$$

which aligns the predicted velocity with the true transport direction from ϵ to $z^{\mathcal{M}'}$. Inference then proceeds as described in section 2.1.

4 Experiments

We extensively evaluate ECHO on a diverse set of dynamical systems from public benchmarks (section 4), with experiments on 2D (section 4.1) and 3D (section 4.2) regular grids and irregular grids (section 4.3). The regular-grid benchmarks allow comparison with many state-of-the-art models that operate directly in physical space and are restricted to uniform meshes, providing evidence for our processor design (i.e. full-trajectory generation). The irregular-grid benchmarks place ECHO in a more demanding regime with complex settings (including partial observations) and varying numbers of mesh points, and are used to compare the efficiency of operator-based approaches that handle arbitrary resolutions. Finally, section 4.4 introduces additional experiment in section F.

Datasets. The 2D and 3D PDE benchmarks span dense regular grids, spherical grids, irregular meshes, parametric dynamics and multi-channel physical fields. Dataset statistics are summarized in Table 6 in the appendix. In particular, we generated high-resolution 2D vorticity trajectories on a dense 1024×1024 grid, with 20 frames per trajectory, yielding more than 20M spatio-temporal values. We also evaluate on large-scale 3D benchmarks from Ohana et al. (2024), including MHD and Turbulence Gravity Cooling, which contain approximately 29M and 25M values per trajectory segment, respectively. Finally, we include Eagle (Janny et al., 2023) and Cylinder Flow (Pfaff et al., 2021) to evaluate ECHO on irregular meshes and non-uniform geometries.

4.1 Evaluation on 2D regular grids

Setting. At inference, ECHO addresses multiple tasks in a zero-shot manner. The comparison is performed on regular grids, where state-of-the-art baselines admit particularly efficient implementations. We compare against neural solvers trained for time-stepping, the standard approach for time-dependent PDEs, on forward and backward prediction. We call *backward prediction* the generation of the earlier part of a trajectory from its last observed frames. This temporal task differs from classical PDE inverse problems, which recover unknown coefficients, forcing terms or initial conditions from observations of the solution. Since the baselines handle one task at a time, we train them separately for each direction, while ECHO handles both without retraining. For the forward (resp. backward) task, we use the first (resp. last) 4 timesteps as context and generate all remaining frames of the trajectory. Regarding the interpolation task, the results are presented in section F.6. All models, including generative ones, are evaluated with a single sample per test trajectory, and we report the relative L^2 error. Probabilistic metrics computed on ensembles are reported in section F.4.

Baselines. We benchmark ECHO against both deterministic and generative baselines. Deterministic models include the transformer-based neural solvers BCAT (Liu et al., 2025), AViT (McCabe et al., 2024), Transolver++ (Luo et al., 2025), and the classical Fourier Neural Operator (FNO). We also evaluate a deterministic variant of ECHO to isolate the gains brought by the generative formulation; this variant, like ECHO, supports multi-task zero-shot inference. As a generative baseline, we consider ENMA (Koupaï et al., 2025), an auto-regressive model trained with a next-token strategy akin to large language models – implementation details are provided in appendix E.

Results Table 1 reports the relative L^2 error across four dynamical systems for both deterministic and generative surrogates. Despite operating in a highly compressed latent space, whereas all baselines except ENMA work directly in physical space without information loss, **ECHO** achieves the best overall accuracy, attaining the lowest error on seven of the eight dataset–task pairs. FNO performs poorly and fails to converge on two of the four datasets. The deterministic version of ECHO remains competitive across benchmarks, while the generative formulation outperforms it on all eight dataset–task pairs, highlighting the benefits of stochastic trajectory modeling. We also report ECHO's performance beyond single-sample accuracy (section F.4). Overall, these results show ECHO's ability to combine high compression with generative modeling, delivering both accuracy and scalability for million-point PDE surrogates. We also evaluate the computation time required to generate a whole trajectory (Latency column in table 1) and show that generative ECHO is on par with the deterministic baselines.

Table 1: Comparison of model performance across four dynamical systems. Errors are relative L^2 errors computed from a single sample per test trajectory, for backward (*Back.*) and forward (*For.*) prediction. Latency is the time needed to generate a whole trajectory, averaged over the datasets. Lower is better for all metrics. The ratio under each dataset name is the compression ratio of the latent models (ECHO, ENMA); the other baselines operate at full resolution ($\times 1$). Best results in **bold**, second best underlined; "–" indicates non-convergence.

		Rayleigh–Bénard ($\times 64$)		Gray–Scott ($\times 32$)		Active Matter ($\times 176$)		ASM ($\times 48$)	
Model	Latency (s)	Back.	For.	Back.	For.	Back.	For.	Back.	For.
Deterministic models									
FNO	**4.42e-2**	2.47e3	4.23e-1	–	–	–	–	1.87	1.52
Transolver++	3.52e-1	6.34e-1	3.31e-1	4.43e-1	2.34e-1	7.33e-1	6.91e-1	1.03	9.64e-1
BCAT	1.18e-1	1.91e-1	1.06e-1	2.19e-1	8.82e-2	4.98e-1	4.56e-1	1.95e-1	2.18e-1
AViT	1.04e-1	4.50e-1	1.01e-1	1.66e-1	7.42e-2	4.50e-1	4.62e-1	**1.04e-1**	1.52e-1
ECHO	4.89e-2	2.53e-1	1.32e-1	8.36e-2	7.66e-2	5.74e-1	3.44e-1	1.18e-1	2.01e-1
Generative models									
ENMA	2.20	1.71e-1	9.87e-2	1.08e-1	5.44e-2	4.27e-1	3.33e-1	1.01	4.12e-1
ECHO	**1.10e-1**	**1.16e-1**	**9.28e-2**	**2.53e-2**	**5.12e-2**	**1.71e-1**	**2.87e-1**	1.12e-1	**1.32e-1**

4.2 Evaluation on 3D regular dynamical systems

To further assess the scalability of ECHO, we provide in table 2 an evaluation on 3D spatio-temporal datasets with trajectory segments respectively with about $29M$ and $25M$ points. The datasets are described in appendix C - table 6.

Table 2: Forward time-stepping on 3D benchmarks. Relative L^2 error over the first frame (1), the first 8 frames (1–8) and the first 16 frames (1–16) of the predicted trajectory. Lower is better. Best results in **bold**, second best underlined; "–" indicates non-convergence.

	MHD			TGC		
Model	1	1–8	1–16	1	1–8	1–16
FNO	**1.69e-1**	–	–	**3.15e-2**	–	–
AViT	3.40e-1	5.68e-1	7.51e-1	1.26e-1	6.22e-1	1.32
BCAT	3.29e-1	5.43e-1	6.70e-1	1.04e-1	9.89e-1	1.62
ECHO	2.70e-1	**3.82e-1**	**5.50e-1**	1.13e-1	**5.55e-1**	**6.67e-1**

Results ECHO successfully scales to 3D scenarios compared to existing baselines. Its full-trajectory generation design accumulates less errors than other baselines and makes it competitive on these datasets. In particular, we observe that FNO performs very well on first frame generation, but quickly diverges as the dynamic evolve, whereas ECHO provides consistent prediction along the entire trajectory.

4.3 Evaluation on irregular meshes

Setting. Following ECHO's success on regular grids (section 4.1), we now consider demanding irregular, dense meshes. We compare arbitrary-resolution neural operators that operate in a compressed latent space and can be queried at arbitrary resolutions. Unless stated otherwise, all methods use matched encoder–decoder compression rates. Each encoder-decoder is first trained and evaluated on reconstruction to assess physical information preservation. Next, to test latent expressivity on a forward task, we fix ECHO as the shared processor due to its superior performance (table 1). All methods thus generate one-shot trajectories (tables 3 and 4). We evaluate two scenarios: fully observed (100% input available) and partially observed (20% randomly sampled points predicting the full grid).

Baselines. We benchmark ECHO against a diverse set of encoders: GINO (Li et al., 2023a; Zhou et al., 2025a) implicit neural representation models CORAL (Serrano et al., 2023); transformer-based auto-encoders AROMA and ENMA (Koupaï et al., 2025); and CALM-PDE (Hagnberger et al., 2025), a continuous-convolution model – details in appendix E.

Table 3: Comparison of encoder–decoder modules from operator surrogates on irregular scenarios. *Rec.* and *For.* denote reconstruction and forward generation; *Irr.* gives the strategy for handling irregular points. $\mathcal{X}_{te}$ is the spatial sampling ratio of the conditioning frames at inference. Colors indicate encoder scalability: green = no issues; yellow = minor modification; red = model size modification required to fit in GPU memory (see section E). Metrics are relative L^2 errors on the full grid of the test set (lower is better). Best results in **bold**, second best underlined.

		Vorticity		Shallow-Water		Eagle		Cylinder Flow	
Irr.	Encoder	Rec.	For.	Rec.	For.	Rec.	For.	Rec.	For.
Full resolution ($\mathcal{X}_{te} = 100\%$)									
Graph	GINO	9.99e-1	1.00	8.68e-1	1.11	5.58e-1	1.89	7.94e-1	8.65e-1
INR	CORAL	5.13e-1	1.34	2.29e-1	6.89e-1	6.04e-1	1.54	2.94e-1	5.08e-1
Attention	AROMA	5.13e-1	8.42e-1	2.51e-2	4.21e-2	2.81e-1	3.09e-1	**2.76e-2**	2.21e-1
	ENMA	4.38e-1	4.38e-1	7.17e-2	7.35e-2	2.85e-1	3.29e-1	7.50e-2	1.04e-1
Convolution	CALM-PDE	2.69e-1	4.20e-1	2.56e-1	2.80e-1	9.01e-1	9.24e-1	2.44e-1	3.56e-1
	ECHO	**6.88e-2**	**1.71e-1**	**1.78e-2**	**1.96e-2**	**1.71e-1**	**2.55e-1**	4.68e-2	**8.82e-2**
Subsampled ($\mathcal{X}_{te} = 20\%$)									
Graph	GINO	9.99e-1	1.00	8.73e-1	1.71	6.77e-1	1.71	7.95e-1	8.65e-1
INR	CORAL	5.19e-1	1.33	2.29e-1	6.87e-1	6.31e-1	1.54	3.31e-1	5.18e-1
Attention	AROMA	5.14e-1	8.43e-1	3.40e-2	4.57e-2	4.01e-1	4.83e-1	**4.59e-2**	2.27e-1
	ENMA	4.41e-1	4.41e-1	8.16e-2	8.09e-1	3.66e-1	**3.99e-1**	8.68e-2	1.20e-1
Convolution	CALM-PDE	**2.95e-1**	4.38e-1	2.77e-1	2.95e-1	9.61e-1	9.75e-1	2.68e-1	3.77e-1
	ECHO	3.66e-1	**2.21e-1**	**2.61e-2**	**2.61e-2**	**2.46e-1**	4.72e-1	5.51e-2	**1.03e-1**

These experiments highlight ECHO's consistent superiority. While graph and INR approaches fail to encode these complex dynamics, attention-based AROMA proves powerful on smaller datasets like *Cylinder Flow* but struggles scaling to *Vorticity* due to high memory costs. Conversely, the convolution-based CALM-PDE handles *Vorticity* better given its lighter footprint. Alongside ECHO, this shows that hierarchical convolutional architectures excel on large-scale data. Furthermore, operating in a spatio-temporally compressed latent space mitigates generation error accumulation. This is evident on *Cylinder Flow*: AROMA yields the best reconstruction, yet ECHO achieves higher forward task accuracy.

Table 4: Super-resolution forecasting on a 1024×1024 *Vorticity* grid (200% of the training grid, more than 20M points). Relative L^2 error of forward generation (*For.*); OOM = out of memory on an 80 GB H100.

Model	GINO	CORAL	AROMA	CALM-PDE	Transolver++	ECHO
For.	OOM	OOM	OOM	OOM	OOM	**3.88e-1**

Large-scale evaluation To further study the scaling properties of ECHO and other baselines, we perform two scaling experiments, pushing the models beyond the capacity of our GPU. First, Table 4 presents results for a spatio-temporal forecasting task on our large-scale *Vorticity* dataset with a grid size of 1024×1024. All baselines ran out of memory on the H100 GPU with 80 GB used for these experiments; only ECHO scaled successfully. Then, Figure 4 shows the memory consumption at inference for several grid sizes. This figure shows the superior scaling capacity of ECHO as it surpasses all baselines in term of memory consumption for grid sizes beyond 100k points and scales up to 100M points (details in section E.1.3).

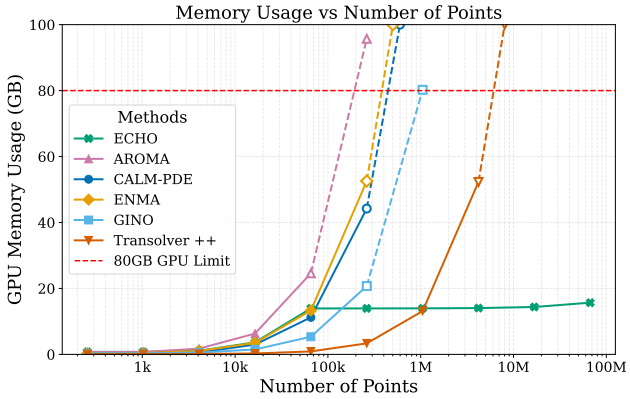


Figure 4: Scaling comparison: memory consumption at inference vs. number of points in the input mesh.

4.4 Additional experiments

Extensive additional experiments are provided in section F. Specifically, we further stress-test ECHO with extreme settings: OOD evaluation on PDE parameters section F.1 - Long range prediction beyond the training horizon section F.2 - Data sampling section F.3 - Probabilistic evaluation section F.4 - Initial Value Problem section F.5 - Interpolation task complete evaluation section F.6 - Additional Evaluation on spatially sparse data appendix F.7. We also provide ablations of key components of ECHO in appendix F.8: hierarchical compression, refinement stage and the generative process. Finally, qualitative samples generated by ECHO are presented in section G.

5 Conclusion and limitations

We introduced ECHO, a generative transformer-based neural operator for large-scale PDE modeling on arbitrary domains. Our central premise is that trajectory-level generative modeling becomes tractable at scale only when performed in strongly compressed yet information-preserving spatio-temporal latent spaces. To this end, ECHO combines hierarchical spatio-temporal compression with latent trajectory generation, enabling efficient modeling of million-point systems while mitigating the error accumulation associated with autoregressive rollouts. Experiments across large-scale 2D and 3D benchmarks demonstrate strong performance on forward prediction, interpolation, backward prediction, and long-horizon forecasting, including simulations with up to 20 million points.

Limitations. The trade-off between compression and reconstruction fidelity is central to this framework and is analyzed empirically in Appendix F.8.1 and fig. 4. More generally, ECHO relies on the existence of compact spatio-temporal latent representations on which trajectory-level generation remains stable and informative. Consequently, extreme compression may affect the recovery of very fine-scale or rapidly evolving structures. Finally, as with any data-driven surrogate model, performance depends on adequate coverage of the underlying data distribution.

6 Acknowledgments

We acknowledge the financial support provided by DL4CLIM (ANR-19-CHIA-0018-01), DEEPNUM (ANR-21-CE23-0017-02), PHLUSIM (ANR-23-CE23-0025-02), MELISSA (ANR-24-CE23-7140-01), PEPR Sharp (ANR-23-PEIA0008, ANR, FRANCE 2030), and cluster IA PostGenAI@Paris (ANR-23-IACL-0007).

This project was provided with computer and storage resources by GENCI at IDRIS thanks to the grants 2025-AD011016890R1, 2025-AD011014938R1, 2026-AD011016890R2, 2026-AD011014938R2, 2025-A0191016890 and 2026-A0211016890 on the supercomputer Jean Zay's V100/A100/H100 partitions.

References

Alkin, B., Fürst, A., Schmid, S., Gruber, L., Holzleitner, M., and Brandstetter, J. Universal physics transformers. *arXiv preprint arXiv:2402.12365*, 2024.

Brandstetter, J., Worrall, D. E., and Welling, M. Message passing neural pde solvers. *International Conference on Learning Representations*, 2022.

Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., and Brown, B. P. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, 2, 2020. ISSN 26431564. doi: 10.1103/PhysRevResearch.2.023068.

Cao, Y., Xu, J., Lin, S., Wei, F., and Hu, H. Global context networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):6881–6895, 2023. doi: 10.1109/TPAMI.2020.3047209.

Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.

Cohen, A. and Devore, R. Approximation of high-dimensional parametric pdes. *Acta Numerica*, 2015.

de Bézenac, E., Pajot, A., and Gallinari, P. Deep learning for physical processes: incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2019, 12 2019. ISSN 17425468. doi: 10.1088/1742-5468/ab3195.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations.*, 10 2021. URL http://arxiv.org/abs/2010.11929.

Elrefaie, M., Morar, F., Dai, A., and Ahmed, F. Drivaernet++: A large-scale multimodal car dataset with computational fluid dynamics simulations and deep learning benchmarks. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 499–536. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/013cf29a9e68e4411d0593040a8a1eb3-Paper-Datasets_and_Benchmarks_Track.pdf.

Esser, P., Rombach, R., and Ommer, B. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., and Rombach, R. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. URL https://openreview.net/forum?id=FPnUhsQJ5B.

Galewsky, J., Scott, R. K., and Polvani, L. M. An initial-value problem for testing numerical models of the global shallow-water equations. *Tellus, Series A: Dynamic Meteorology and Oceanography*, 56, 2004. ISSN 02806495. doi: 10.1111/j.1600-0870.2004.00071.x.

Gordon, J., Bruinsma, W. P., Foong, A. Y. K., Requeima, J., Dubois, Y., and Turner, R. E. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2020.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062, 2021.

Hagnberger, J., Musekamp, D., and Niepert, M. CALM-PDE: Continuous and adaptive convolutions for latent space modeling of time-dependent PDEs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0r4yzkvt9j.

Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., and Zhu, J. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023.

Hao, Z., Su, C., Liu, S., Berner, J., Ying, C., Su, H., Anandkumar, A., Song, J., and Zhu, J. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *41th International Conference on Machine Learning (ICML 2024)*, 2024.

Herde, M., Raonić, B., Rohner, T., Käppeli, R., Molinaro, R., de Bézenac, E., and Mishra, S. Poseidon: Efficient foundation models for pdes. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 72525–72624. Curran Associates, Inc., 2024. doi: 10.52202/079017-2311. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/84e1b1ec17bb11c57234e96433022a9a-Paper-Conference.pdf.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Holzschuh, B., Liu, Q., Kohl, G., and Thuerey, N. PDE-transformer: Efficient and versatile transformers for physics simulations. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=3BaJMRaPSx.

Huang, J., Yang, G., Wang, Z., and Park, J. J. Diffusionpde: Generative pde-solving under partial observation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Janny, S., Benetteau, A., Nadri, M., Digne, J., Thome, N., and Wolf, C. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Kohl, G., Chen, L.-W., and Thuerey, N. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation, 2024. URL https://arxiv.org/abs/2309.01745.

Kossaifi, J., Kovachki, N., Li, Z., Pitt, D., Liu-Schiaffini, M., Duruisseaux, V., George, R. J., Bonev, B., Azizzadenesheli, K., Berner, J., and Anandkumar, A. A library for learning neural operators. *arXiv preprint arXiv:2412.10354*, 2025.

Koupaï, A. K., Benet, J. M., Yin, Y., Vittaut, J.-N., and Gallinari, P. Geps: Boosting generalization in parametric pde neural solvers through adaptive conditioning. *Advances in Neural Information Processing Systems*, 38, 2024.

Koupaï, A. K., Boudec, L. L., Serrano, L., and Gallinari, P. ENMA: Tokenwise autoregression for continuous neural PDE operators. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=3CYXSMFv55.

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. URL http://jmlr.org/papers/v24/21-1524.html.

Li, E., Wang, Z., Huang, J., and Park, J. J. Videopde: Unified generative pde solving via video. *arXiV*, 2025a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *International Conference on Learning Representations.*, 10 2021. URL http://arxiv.org/abs/2010.08895.

Li, Z., Kovachki, N., Choy, C., Li, B., Kossaifi, J., Otta, S., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., and Anandkumar, A. Geometry-informed neural operator for large-scale 3d pdes. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 35836–35854. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/70518ea42831f02afc3a2828993935ad-Paper-Conference.pdf.

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856. URL https://openreview.net/forum?id=EPPqt3uERT.

Li, Z., Zhou, A., and Farimani, A. B. Generative latent neural pde solver using flow matching. *arXiv preprint arXiv:2503.22600*, 2025b. URL http://arxiv.org/abs/2503.22600.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T. Q., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.

Lippe, P., Veeling, B. S., Perdikaris, P., Turner, R. E., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. In *Neurips*, pp. 1–36, 2023. URL http://arxiv.org/abs/2308.05732.

Liu, Y., Sun, J., and Schaeffer, H. BCAT: A block causal transformer for pde foundation models for fluid dynamics. *arXiv preprint arXiv:2501.18972*, 2025.

Long, Z., Lu, Y., Ma, X., and Dong, B. PDE-net: Learning PDEs from data. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3208–3216. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/long18a.html.

Lu, L., Jin, P., and Karniadakis, G. E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nat Mach Intell*, 3:218–229, 10 2021. doi: 10.1038/s42256-021-00302-5. URL http://arxiv.org/abs/1910.03193http://dx.doi.org/10.1038/s42256-021-00302-5.

Luo, H., Wu, H., Zhou, H., Xing, L., Di, Y., Wang, J., and Long, M. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=AM7iAh0krx.

McCabe, M., Blancard, B. R.-S., Parker, L. H., Ohana, R., Cranmer, M., Bietti, A., Eickenberg, M., Golkar, S., Krawezik, G., Lanusse, F., Pettee, M., Tesileanu, T., Cho, K., and Ho, S. Multiple physics pretraining for spatiotemporal surrogate models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DKSI3bULiZ.

Mousavi, S., Wen, S., Lingsch, L., Herde, M., Raonic, B., and Mishra, S. RIGNO: A graph-based framework for robust and accurate operator learning for PDEs on arbitrary domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ahJfROJOYt.

Nguyen, T., Koneru, A., Li, S., et al. Physix: A foundation model for physics simulations. *arXiv preprint arXiv:2506.17774*, 2025.

Ohana, R., McCabe, M., Meyer, L., Morel, R., Agocs, F., Beneitez, M., Berger, M., Burkhart, B., Dalziel, S., Fielding, D., et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pearson, J. E. Complex patterns in a simple system. *Science*, 1993. doi: 10.1126/science.261.5118.189.

Pedersen, C., Zanna, L., and Bruna, J. Thermalizer: Stable autoregressive neural emulation of spatiotemporal chaos. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=73zrUyI5kB.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *International Conference on Learning Representations.*, 10 2021. URL http://arxiv.org/abs/2010.03409.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2 2019. ISSN 10902716. doi: 10.1016/j.jcp.2018.10.045.

Rozet, F., Ohana, R., McCabe, M., Louppe, G., Lanusse, F., and Ho, S. Lost in latent space: An empirical study of latent diffusion models for physics emulation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=xoNrbfbekM.

Serrano, L., Boudec, L. L., Koupaï, A. K., Wang, T. X., Yin, Y., Vittaut, J.-N., and Gallinari, P. Operator learning with neural fields: Tackling pdes on general geometries. *Neural Information Processing Systems*, 2023.

Serrano, L., Wang, T. X., Le Naour, E., Vittaut, J.-N., and Gallinari, P. Aroma: Preserving spatial structure for latent pde modeling with local neural fields. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 13489–13521. Curran Associates, Inc., 2024. doi: 10.52202/079017-0431. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/185a120a3f709187e68bd092e6098851-Paper-Conference.pdf.

Serrano, L., Koupaï, A. K., Wang, T. X., ERBACHER, P., and Gallinari, P. Zebra: In-context generative pretraining for solving parametric PDEs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=22kNOkkokU.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tmIiMPl4IPa.

van den Oord, A., Vinyals, O., and kavukcuoglu, k. Neural discrete representation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.

Wang, H., Song, S., and Huang, G. Gridmix: Exploring spatial modulation for neural fields in PDE modeling. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=FurODtynPX.

Wang, S., Seidman, J. H., Sankaran, S., Wang, H., Pappas, G. J., and Perdikaris, P. CVit: Continuous vision transformer for operator learning. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=cRnCcuLvyr.

Wang, T. and Wang, C. Latent neural operator for solving forward and inverse pde problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Wen, S., Kumbhat, A., Lingsch, L., Mousavi, S., Zhao, Y., Chandrashekar, P., and Mishra, S. Geometry aware operator transformer as an efficient and accurate neural surrogate for PDEs on arbitrary domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=HXFvNkNt0n.

Wu, H., Luo, H., Wang, H., Wang, J., and Long, M. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning*, 2024.

Yin, Y., Kirchmeyer, M., Franceschi, J.-Y., Rakotomamonjy, A., and Gallinari, P. Continuous pde dynamics forecasting with implicit neural representations. *International Conference on Learning Representations*, 9 2022. URL http://arxiv.org/abs/2209.14855.

Yu, L., Cheng, Y., Sohn, K., Lezama, J., Zhang, H., Chang, H., Hauptmann, A. G., Yang, M.-H., Hao, Y., Essa, I., et al. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10459–10469, 2023.

Yu, L., Lezama, J., Gundavarapu, N. B., Versari, L., Sohn, K., Minnen, D., Cheng, Y., Gupta, A., Gu, X., Hauptmann, A. G., Gong, B., Yang, M.-H., Essa, I., Ross, D. A., and Jiang, L. Language model beats diffusion - tokenizer is key to visual generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gzqrANCF4g.

Zhou, A., Li, Z., Schneier, M., Jr, J. R. B., and Farimani, A. B. Text2PDE: Latent diffusion models for accessible physics simulation. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=Nb3a8aUGfj.

Zhou, H., Ma, Y., Wu, H., Wang, H., and Long, M. Unisolver: PDE-conditional transformers towards universal neural PDE solvers. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=r1ryQoI9iZ.

Appendix: Table of Contents

A **Statements** 18
A.1 Impact Statement 18
A.2 Reproductibility Statement 18
A.3 Use of LLMs 18
B **Related works** 19
B.1 Operator learning 19
B.2 Generative models 20
C **Dataset details** 21
C.1 Vorticity 21
C.2 Shallow-water 22
C.3 Eagle 22
C.4 Cylinder-flow 23
C.5 Rayleigh-Bénard 23
C.6 Gray-Scott 24
C.7 Active matter 25
C.8 Acoustic scattering maze 27
C.9 Magnetohydrodynamic (MHD) 27
C.10 Turbulence gravity cooling (TGC) 28
D **Architecture details** 29
D.1 Inference model 29
D.2 Encoder-decoder 29
D.2.1 Interpolation with continuous convolution 29
D.2.2 Spatial and temporal compression 31
E **Implementation details** 32
E.1 Evaluation protocol of the process 32
E.1.1 Baseline details 32
E.1.2 Training details 32
E.1.3 Fairness of Baseline Comparisons and Scaling Limits. 33
E.2 Encoder-decoder implementation protocol 35
E.2.1 ECHO encoder-decoder architecture 35
E.2.2 Baseline details 35
E.2.3 Training details 36
F **Additional experiments** 38
F.1 Out-of-distribution evaluation 38
F.2 Long-range prediction 39
F.2.1 Long-range prediction on Vorticity 39
F.2.2 Long-range prediction on Shallow-Water 39
F.2.3 Spectral analysis on the vorticity generation 39
F.3 Generative data sampling 40
F.4 Probabilistic evaluation 40
F.5 Initial value problem 42
F.6 Interpolation tasks 42
F.7 Encoding sparse data 42
F.8 Ablation studies 44
F.8.1 Impact of hierarchical compression 44
F.8.2 Impact of the refinement stage 44
F.8.3 Ablations on the generative process 44
G **Visualizations** 46

G.1 Vorticity 46
G.2 Gray-Scott 47
G.3 Rayleigh-Bénard 48
G.4 Active-Matter 49
G.5 Acoustic Scattering Maze 50
G.6 Eagle 51

A Statements

A.1 Impact Statement

PDEs are involved in many applications of science and engineering. This paper introduces a new method to improve the performance and scalability of surrogate models for solving PDEs. While we do not directly target such real-world applications in this paper, one should acknowledge that solvers can be used in a wide range of scenarios including weather, climate, medical, aerodynamics, industry, and military applications.

A.2 Reproducibility Statement

We provide implementation details in section E including architecture details, training hyper-parameters and baseline descriptions. Moreover, we will publicly release the code base and the generated dataset upon acceptance. All our experiments run on Nvidia H100 GPUs with 80Gb memory. We estimate the total compute budget for the development of ECHO to be approximately of 50 days of compute.

A.3 Use of LLMs

LLMs were used for polishing the text formulation of this publication.

B Related Works

Table 5: Comparison of operator learning approaches for spatio-temporal PDE forecasting.

Model	Reference	Arbitrary domains	Compression	High-res. encoding	Hierarchical encoding	Generative modeling	Multi-task prediction
Graph							
MPPDE	Brandstetter et al. (2022)	✓	✗	✗	✗	✗	✗
RIGNO	Mousavi et al. (2025)	✓	✓	✗	✗	✗	✗
Operator							
Transolver++	Luo et al. (2025)	✓	✗	✓	✗	✗	✗
GINO	Li et al. (2023a)	✓	✓	✗	✗	✗	✗
TEXT2PDE	Zhou et al. (2025a)	✓	✓	✗	✓	✓	✗
INRs							
CORAL	Serrano et al. (2023)	✓	✓	✗	✗	✗	✗
DiNo	Yin et al. (2022)	✓	✓	✗	✗	✗	✗
Attention							
UPT	Alkin et al. (2024)	✓	✓	✗	✗	✗	✗
ENMA	Koupaï et al. (2025)	✓	✓	✗	✓	✓	✗
Convolution							
CALM-PDE	Hagnberger et al. (2025)	✓	✓	✗	✓	✗	✗
ECHO	Ours	✓	✓	✓	✓	✓	✓

B.1 Operator Learning

Operator learning has emerged as a powerful paradigm for modeling mappings between infinite-dimensional function spaces, enabling neural surrogates to learn the solution operators of partial differential equations (PDEs) directly. Early breakthroughs such as DeepONet and the Fourier Neural Operator (FNO) established neural operators (NOs) as effective and mesh-independent tools for approximating PDE solution maps (Li et al., 2021; Lu et al., 2021). Since then, research has focused on improving both the expressiveness and scalability of NOs. For example, multi-scale modeling has been explored through factorized representations and wavelet-based decompositions (Gupta et al., 2021), while latent-space formulations have been introduced to support complex and irregular geometries (Tran et al., 2023; Li et al., 2023a). Implicit neural representations (INRs), as in CORAL, have also been proposed to flexibly handle variable spatial discretizations at inference time (Yin et al., 2022; Serrano et al., 2023; Wang et al., 2025a).

A major trend in recent years has been the adoption of *attention-based architectures* for operator learning on irregular meshes. OFormer introduced transformers for embedding input–output function pairs, demonstrating their strong representation capabilities for operator approximation (Li et al., 2023b). This line of work inspired more advanced designs such as GNOT and Transolver, which refine input encoding and generalization to complex domains (Hao et al., 2023; Wu et al., 2024). However, transformer-based NOs often suffer from quadratic complexity and memory usage, making them difficult to scale to real-world scenarios where spatial resolutions can reach millions of points.

These limitations have motivated the development of *efficient neural operator surrogates* aimed at reducing computational and memory costs without sacrificing accuracy. Perceiver-inspired approaches such as Aroma and UPT learn compact latent representations that decouple the number of tokens from the raw input size, significantly improving scalability (Serrano et al., 2024; Alkin et al., 2024). While these methods achieve strong results on moderately large inputs, our empirical analysis in fig. 1 (left) shows that attention-only latent encoders degrade at extreme compression ratios, limiting their effectiveness for high-resolution generative modeling.

A complementary direction has explored *hierarchical compression*. CALM-PDE (Hagnberger et al., 2025) demonstrated that progressively reducing spatial resolution via continuous convolutions improves efficiency while preserving fidelity. Building on this idea, we introduce a spatio-temporal hierarchical encoder that extends this principle beyond static fields to dense time-dependent trajectories. By gradually compressing both space and time from a dense latent grid to coarser levels,

our approach constructs highly informative and compact latent spaces, enabling efficient generative modeling even on million-point spatio-temporal inputs. The continuous convolution that maps irregular inputs onto a regular latent grid is closely related to the CONVDEEPSET layer of Convolutional Conditional Neural Processes (Gordon et al., 2020), which also aggregates off-the-grid observations onto a grid permutation-invariantly through a learned kernel.

B.2 Generative Models

While most operator learning methods are deterministic, generative modeling introduces key capabilities for physical systems—most notably the ability to capture uncertainty and represent one-to-many mappings. This is especially valuable in chaotic or partially observed regimes, where deterministic predictions can quickly diverge. Two main generative paradigms have emerged for scientific modeling: diffusion-based methods and autoregressive transformers.

Diffusion Transformers. Diffusion models synthesize data by learning to reverse a progressive noising process through a sequence of denoising steps (Ho et al., 2020). In computer vision, Latent Diffusion Transformers (DiTs) extend this approach by operating in the latent space of a variational autoencoder (VAE), achieving strong generative quality and scalability (Peebles & Xie, 2022).

Recently, diffusion has been adapted to physical modeling. Kohl et al. (2024) introduced an autoregressive diffusion framework for PDE dynamics, aiming to better capture stochasticity in turbulent flows. Lippe et al. (2023) improved the treatment of chaotic high-frequency regimes by modulating noise variance during denoising. Generative diffusion is particularly attractive under partial observations or incomplete physics (Huang et al., 2024). For example, Li et al. (2025a) reformulate PDE solving as a video inpainting problem to handle both forward and backward tasks, but the method remains limited to regular grids and is expensive at inference. Zhou et al. (2025a) further extend DiTs to physics-driven data generation from textual prompts.

Despite their flexibility, diffusion models are computationally heavy at inference due to the large number of denoising steps. *Flow Matching* provides a more efficient alternative by learning a continuous-time velocity field that transports one distribution to another via an ordinary differential equation (ODE), enabling much faster sampling with far fewer steps (Lipman et al., 2023, 2024). Flow matching has recently been applied to PDE surrogates (Holzschuh et al., 2025), mostly in multi-physics settings and under fixed, regular discretizations. More recently, (Li et al., 2025b), proposed to use Flow matching to solve PDE framewise. The main difference with our proposed method lies in the formulation that we adopted, that allows us to solve several tasks without retraining. Moreover, ECHO makes use of continuous convolution, allowing our model to solve PDE on very dense grids.

Autoregressive Transformers. Autoregressive (AR) models, originally designed for language modeling, have been extended to visual domains by sequentially generating spatial and temporal tokens. A common design couples a vector-quantized variational autoencoder (VQ-VAE) with a causal or bidirectional transformer to model discrete token sequences (van den Oord et al., 2017; Esser et al., 2021; Chang et al., 2022). In video generation, *Magvit* and *Magvit2* (Yu et al., 2023, 2024) use 3D CNNs to encode spatiotemporal structure and autoregressively predict quantized latent tokens for future frames.

This paradigm has recently been adapted to PDE surrogates. *Zebra* (Serrano et al., 2025) combines a spatial VQ-VAE with a causal transformer for in-context forecasting of dynamical systems. However, reliance on discrete codebooks can limit expressiveness and hinder accurate representation of fine-grained continuous physical phenomena. Nguyen et al. (2025) extend this design to multi-physics settings by adding a decoder refinement stage to reduce quantization artifacts, at the cost of extra complexity. Alternatively, Koupaï et al. (2025) replace discrete tokens with continuous ones and train next-token generation via a flow-matching objective.

Yet, whether discrete or continuous, autoregressive strategies remain prone to long-horizon error accumulation, particularly in compressed latent spaces where information loss amplifies drift from the true solution (Pedersen et al., 2025). Our work departs from the next-step paradigm: we train a flow-matching transformer to synthesize entire trajectory segments directly rather than predicting one token at a time. This design mitigates temporal drift and enables more stable long-range rollouts than both diffusion-based and deterministic baselines (see the middle panel of fig. 1).

C Dataset details

The following section details the datasets used in the experiments. They are first summarized in table 6 and then, we detail and illustrate each dataset.

Table 6: Datasets used in the experiments. Sizes are given as $T \times (H \times W \times D) \times C$, where the depth D only applies to 3D datasets. ASM = Acoustic Scattering Maze; MHD = Magnetohydrodynamics; TGC = Turbulence Gravity Cooling; BC = boundary conditions.

Dataset	# points	Grid	PDE parameters	Size $T \times (H \times W \times D) \times C$
2D				
Vorticity (Koupaï et al., 2024)	20M	Dense	Parametric	$20 \times \mathbf{1048576} \times 1$
Shallow-Water (Yin et al., 2022)	2.6M	Spherical	Single instance	$40 \times 32768 \times 2$
Gray–Scott (Ohana et al., 2024)	1.3M	–	Varying	$40 \times 16384 \times 2$
Rayleigh–Bénard (Ohana et al., 2024)	3.9M	–	Varying	$15 \times 65536 \times \mathbf{4}$
ASM (Ohana et al., 2024)	2.9M	–	Varying	$15 \times 65536 \times \mathbf{3}$
Active Matter (Ohana et al., 2024)	14.4M	–	Varying	$20 \times 65536 \times \mathbf{11}$
Eagle (Janny et al., 2023)	0.2M	Irregular	BC	$20 \times 2500 \times 4$
Cylinder Flow (Pfaff et al., 2021)	0.3M	Irregular	Single instance	$60 \times 2000 \times 3$
3D				
MHD (Ohana et al., 2024)	29M	–	Varying	$16 \times 262144 \times \mathbf{7}$
TGC (Ohana et al., 2024)	25M	–	Varying	$16 \times 262144 \times \mathbf{6}$

C.1 Vorticity

We consider a 2D turbulence model and focus on the evolution of the vorticity field ω, which captures the local rotation of the fluid and is defined as $\omega = \nabla \times \mathbf{u}$, where $\mathbf{u}$ is the velocity field. The governing equation is:

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega - \nu \nabla^2 \omega = 0, \tag{10}$$

where ν denotes the kinematic viscosity, defined as $\nu = 1/\text{Re}$. We sample $\nu \sim \mathcal{U}([10^{-3}, 10^{-2}])$. The initial conditions are generated from the energy spectrum:

$$E(k) = \frac{4}{3}\sqrt{\pi}\left(\frac{k}{k_0}\right)^4 \frac{1}{k_0} \exp\left(-\left(\frac{k}{k_0}\right)^2\right), \tag{11}$$

where k_0 denotes the characteristic wavenumber. Vorticity is linked to energy by the following equation :

$$\omega(k) = \sqrt{\frac{E(k)}{\pi k}} \tag{12}$$

We generate very dense-grid with $1,024 \times 1,024$ spatial resolution and 20 timesteps, resulting in trajectories with more than 20M points.

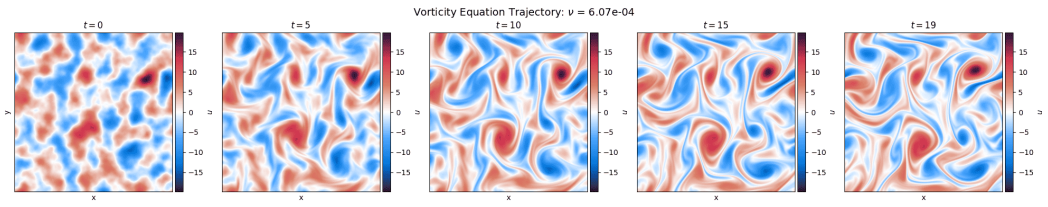


Figure 5: Sample of the Vorticity dataset.

C.2 Shallow-Water

3D-Spherical Shallow-Water (*Shallow-Water*). We consider the shallow-water equations on a rotating sphere, modeling large-scale atmospheric flows:

$$\frac{\partial u}{\partial t} = -f \cdot k \times u - g\nabla h + \nu \Delta u, \tag{13}$$

$$\frac{\partial h}{\partial t} = -h\nabla \cdot u + \nu \Delta h, \tag{14}$$

where k is the outward unit normal to the sphere, u is the tangential velocity field (with vorticity $w = \nabla \times u$), and h denotes the fluid height.

Data is generated with the *Dedalus* software (Burns et al., 2020), following Yin et al. (2022). The setup produces symmetric dynamics in both hemispheres. The initial zonal velocity u_0 contains two symmetric jets parallel to latitude circles:

$$u_0(\phi, \theta) = \begin{cases} \left(\frac{u_{max}}{e_n} \exp\left(\frac{1}{(\phi-\phi_0)(\phi-\phi_1)}\right), 0\right), & \phi \in (\phi_0, \phi_1), \\ \left(\frac{u_{max}}{e_n} \exp\left(\frac{1}{(\phi+\phi_0)(\phi+\phi_1)}\right), 0\right), & \phi \in (-\phi_1, -\phi_0), \\ (0, 0), & \text{otherwise}, \end{cases} \tag{15}$$

where ϕ and θ are latitude and longitude, u_{max} is the maximum velocity, $\phi_0 = \frac{\pi}{7}$, $\phi_1 = \frac{\pi}{2} - \phi_0$, and $e_n = \exp\left(-\frac{4}{(\phi_1-\phi_0)^2}\right)$.

The initial fluid height h_0 is computed by solving a boundary value problem as in Galewsky et al. (2004), perturbed with:

$$h_0'(\phi, \theta) = \hat{h} \cos(\phi) \exp\left(-\left(\frac{\theta}{\alpha}\right)^2\right) \left[\exp\left(-\left(\frac{\phi_2 - \phi}{\beta}\right)^2\right) + \exp\left(-\left(\frac{\phi_2 + \phi}{\beta}\right)^2\right)\right], \tag{16}$$

with constants $\phi_2 = \frac{\pi}{4}$, $\hat{h} = 120$ m, $\alpha = \frac{1}{3}$, and $\beta = \frac{1}{15}$ (Galewsky et al., 2004). Simulations are run on a latitude–longitude grid of size 128×256.

For data generation, u_{max} is sampled from $\mathcal{U}(60, 80)$. Trajectories span 320 hours with hourly snapshots, yielding 320 frames each. To emphasize meaningful dynamics, we retain only the last 160 frames. Each long trajectory is split into sub-trajectories of 40 frames. The resulting dataset comprises 64 training trajectories and 8 test trajectories. Finally, we rescale the fields for numerical stability: heights h by 3×10^3 and vorticity w by 2.

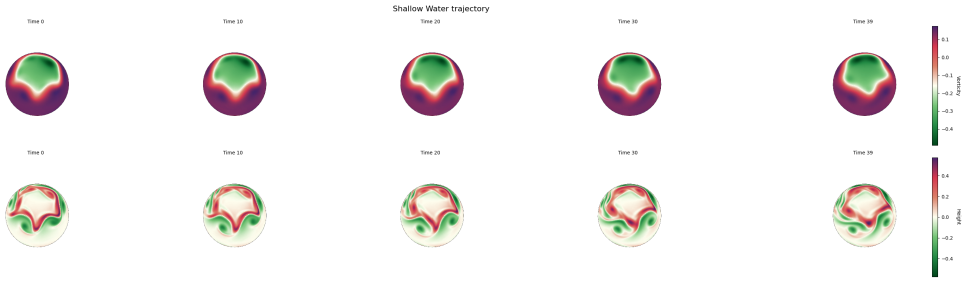


Figure 6: Sample of the Shallow-water dataset.

C.3 Eagle

Eagle is taken from (Janny et al., 2023) and represents the fluid velocity and pressure around a moving source. This dataset has different boundary shapes that create varying flow around the source. Moreover, trajectories are long (990 timesteps), making this dataset well-suited for long-rollout tasks. For additional deatails on dataset generation, we refer readers to (Janny et al., 2023). We provide a visualization of a trajectory in fig. 7

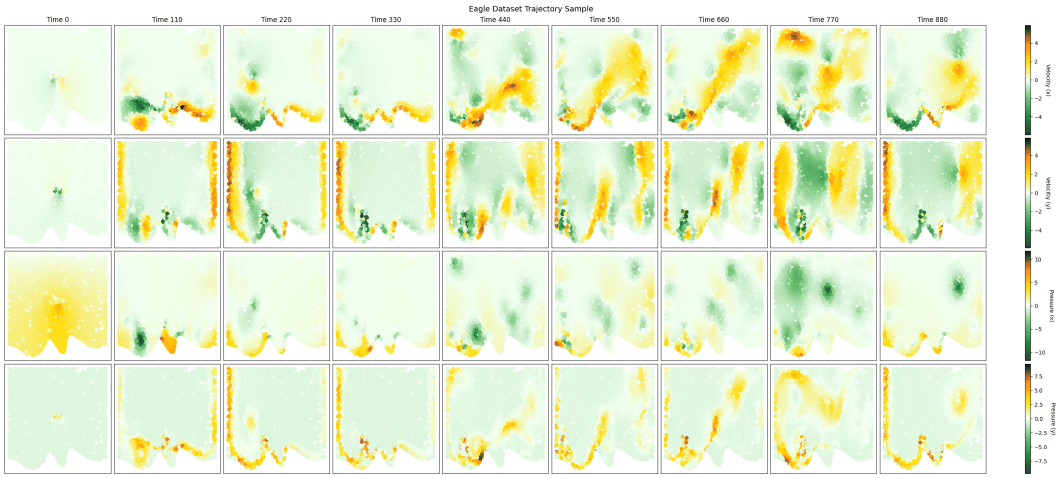


Figure 7: Sample of the Eagle dataset.

C.4 Cylinder-Flow

We use the dataset introduced by Pfaff et al. (2021), which simulates water flow in a channel with a cylinder serving as an obstacle. The governing equations are the 2D incompressible Navier–Stokes equations with constant density:

$$\partial_t \mathbf{v} = 0, \tag{17}$$

$$\rho_0 \left(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}\right) + \nabla p = \mu \nabla^2 \mathbf{v}, \tag{18}$$

$$\mathbf{v} := \mathbf{v}(t, \omega), \quad p := p(t, \omega), \quad \omega \in \Omega,\, t \in [0, T], \tag{19}$$

where ρ_0 denotes the constant density, $\mathbf{v}$ the velocity field, and p the pressure. Each sample varies in the cylinder's diameter and position, and the task of neural surrogates is to predict both velocity and pressure fields. For our experiments, we subsample trajectories to 15 timesteps. Further details about the setup can be found in Pfaff et al. (2021).

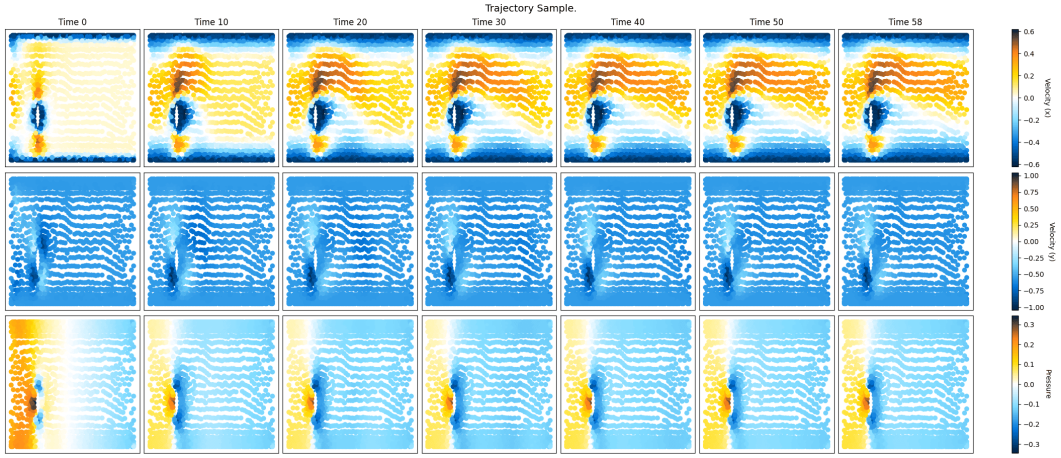


Figure 8: Sample of the Cylinder Flow dataset.

C.5 Rayleigh-Bénard

We consider a 2D horizontally periodic fluid subject to buoyancy-driven convection. The dataset is taken from (Ohana et al., 2024), which provides the full setting for data generation. The state variables are the velocity $u = (u_x, u_z)$, buoyancy b, and pressure p. Heating from below and cooling from above create density variations that drive convection, producing characteristic Bénard cells where hot fluid rises and cold fluid sinks.

The dynamics are governed by

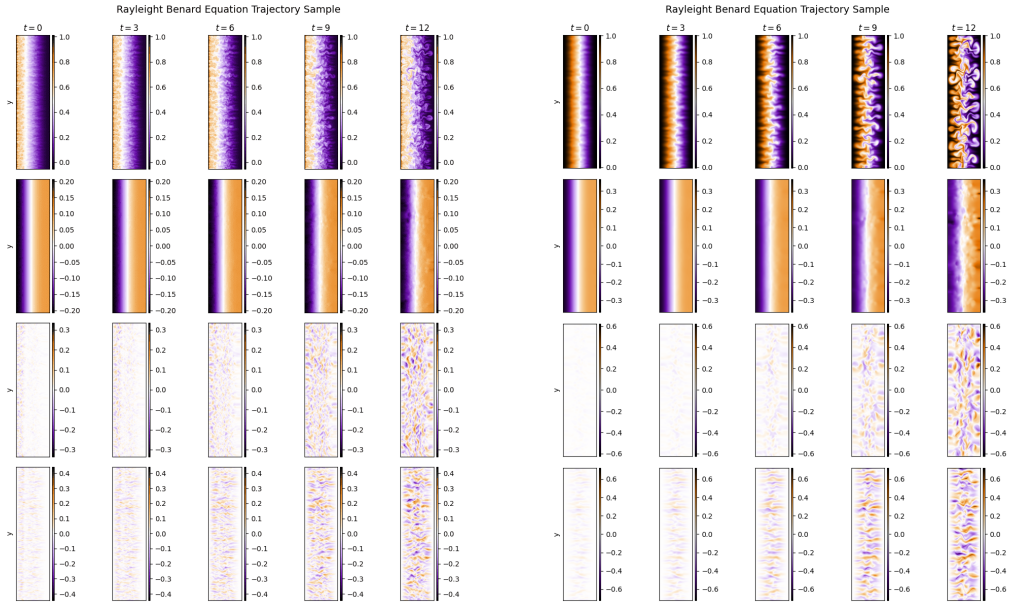
$$\frac{\partial b}{\partial t} - \kappa \Delta b = -u \cdot \nabla b, \tag{20}$$

$$\frac{\partial u}{\partial t} - \nu \Delta u + \nabla p - b e_z = -u \cdot \nabla u, \tag{21}$$

with $\Delta = \nabla \cdot \nabla$ the Laplacian, e_z the vertical unit vector, and $\int p = 0$ a gauge constraint. The first equation describes convection–diffusion of buoyancy, while the second is the Navier–Stokes equation with buoyancy forcing. The domain is periodic in x with vertical boundary conditions $u(z = 0) = u(z = L_z) = 0$ and $b(z = 0) = b(z = L_z) = 0$. The system is parameterized by the Rayleigh and Prandtl numbers through thermal diffusivity κ and viscosity ν:

$$\kappa = (\text{Rayleigh} \times \text{Prandtl})^{-1/2}, \quad \nu = \left(\frac{\text{Rayleigh}}{\text{Prandtl}}\right)^{-1/2}. \tag{22}$$

Here, the Rayleigh number measures the ratio of buoyancy to diffusion forces, while the Prandtl number controls the balance between momentum and thermal diffusion. We provide in fig. 9 visualizations of 2 trajectories.



(a) (b)

Figure 9: Samples from the *Rayleigh-Bénard* dataset.

C.6 Gray-Scott

The Gray–Scott system (Pearson, 1993) is a pair of coupled reaction–diffusion equations modeling the interaction of two chemical species, A and B, whose concentrations evolve in space and time:

$$\frac{\partial A}{\partial t} = \delta_A \Delta A - AB^2 + f(1 - A), \tag{23}$$

$$\frac{\partial B}{\partial t} = \delta_B \Delta B + AB^2 - (f + k)B. \tag{24}$$

Here, f and k are the *feed* and *kill* rates: f controls the supply of species A, while k regulates the removal of species B. The diffusion coefficients δ_A and δ_B determine the transport rates of the two species.

The dataset used in our experiments is provided by Ohana et al. (2024), which includes further details on its generation process.See fig. 10 for a visualization of a trajectory.

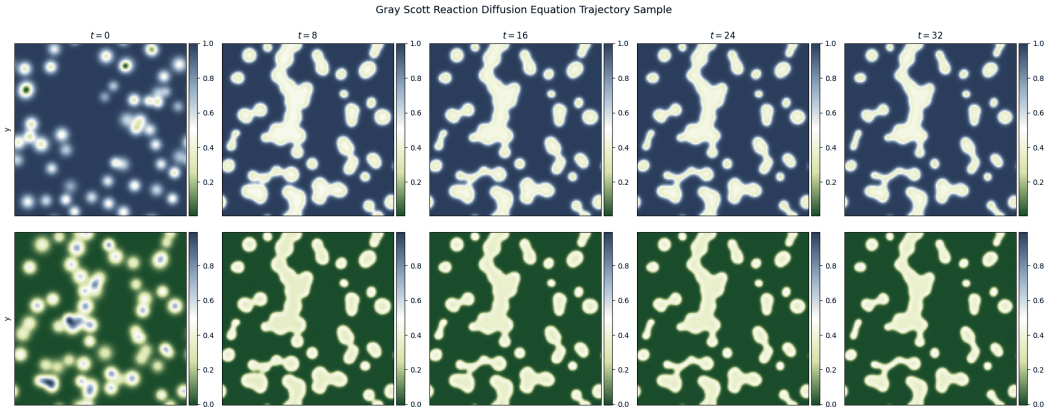


Figure 10: Sample of the *Gray-Scott Reaction-Diffusion* dataset

C.7 Active Matter

Active Matter features simulations of rod-like biological active particles immersed in a Stokes flow. The active particles transfer chemical energy into mechanical work, leading to stresses that are communicated across the system. Furthermore, particle coordination causes complex behavior inside the flow. More details are provided in (Ohana et al., 2024) for the dataset generation setup.We provide a visualization of a trajectory in fig. 11.

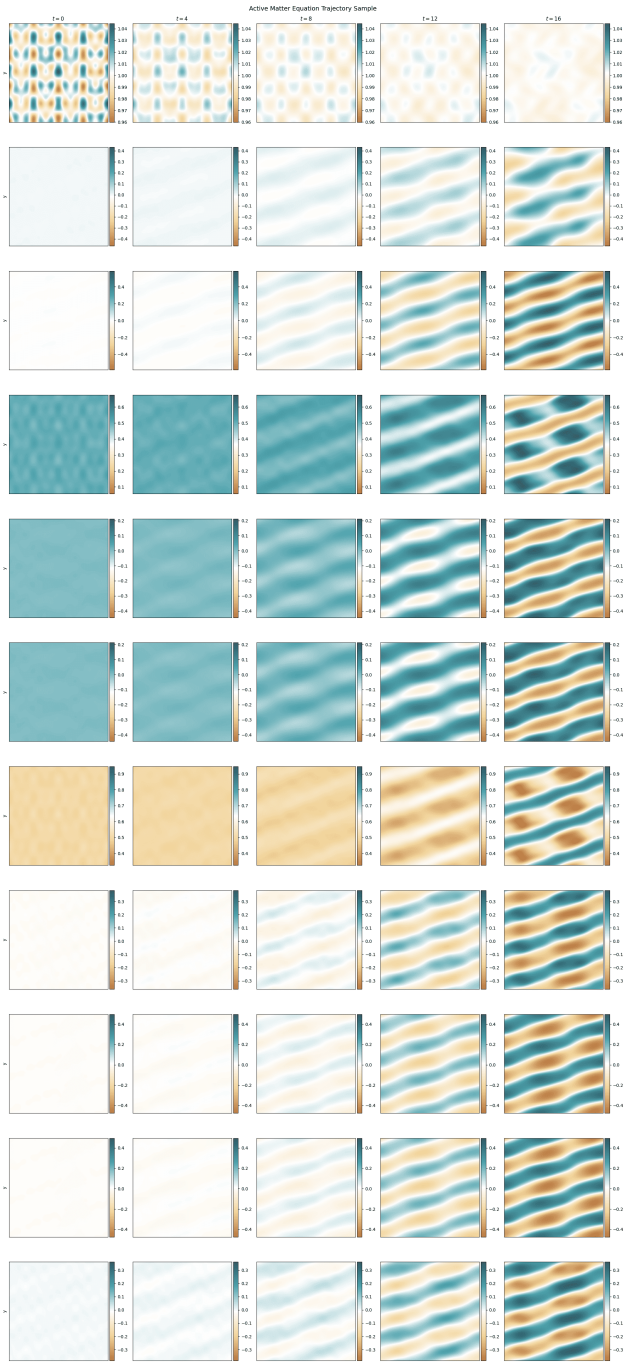


Figure 11: Trajectory sample of *Active Matter* dataset.

C.8 Acoustic Scattering Maze

We use the *Acoustic Scattering Maze* dataset introduced by Ohana et al. (2024). The dynamics are governed by acoustic equations that describe the propagation of a pressure wave through materials with spatially varying density. The governing system is:

$$\frac{\partial p}{\partial t} + K(x,y)\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \tag{25}$$

$$\frac{\partial u}{\partial t} + \frac{1}{\rho(x,y)}\frac{\partial p}{\partial x} = 0 \tag{26}$$

$$\frac{\partial v}{\partial t} + \frac{1}{\rho(x,y)}\frac{\partial p}{\partial y} = 0 \tag{27}$$

where ρ denotes the material density, u and v the velocity components in the x and y directions, p the pressure, and K the bulk modulus. Together, ρ and K determine the speed of sound; in these simulations, ρ varies spatially while K is fixed to a constant value of 4.

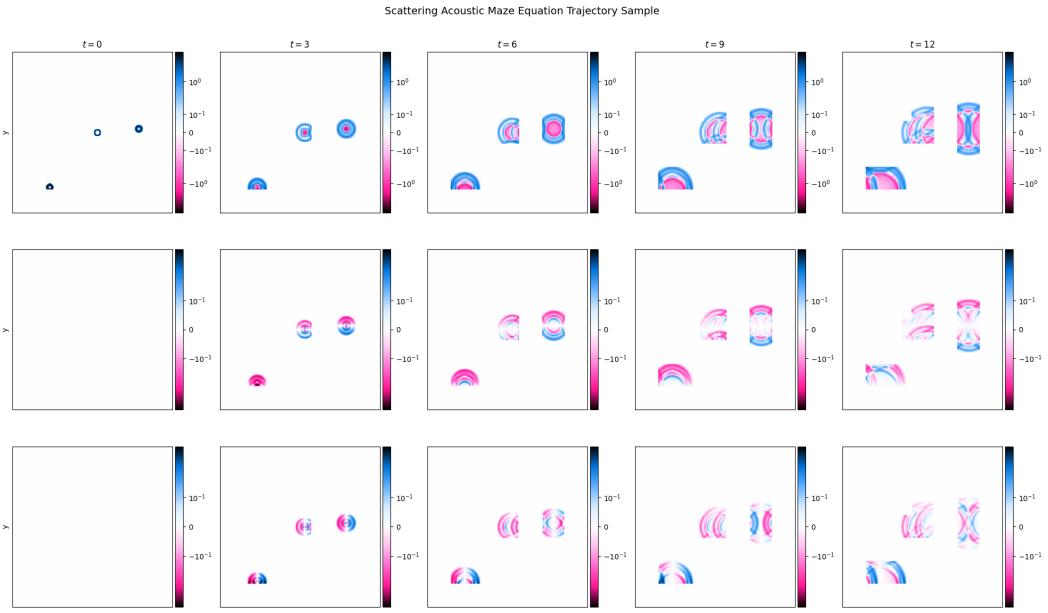
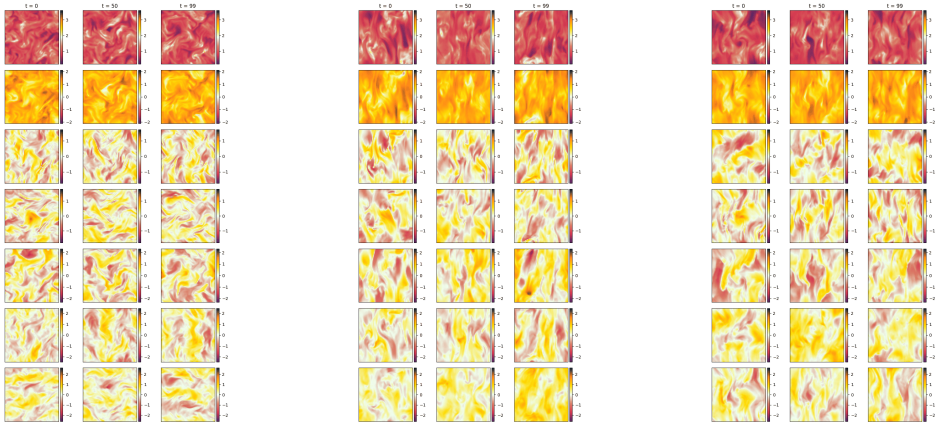


Figure 12: Sample of the *Acoustic Scattering Maze* dataset.

C.9 Magnetohydrodynamic (MHD)

The MHD dataset (Ohana et al., 2024), simulates a magnetohydrodynamic (MHD) turbulence. Such dynamics are often used to describe space events such as solar winds or galaxy formation. We refer to (Ohana et al., 2024) for additional details. We provide in fig. 13 a trajectory sample.

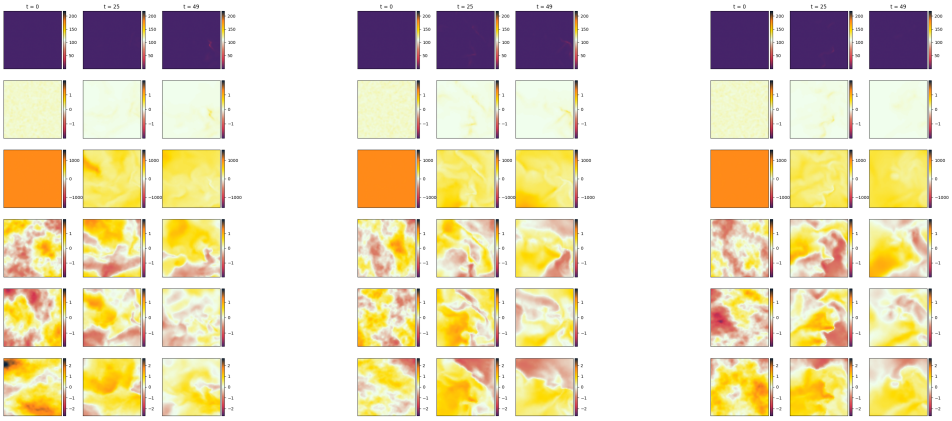


(a) Slice along x. (b) Slice along y. (c) Slice along z.

Figure 13: Trajectory sample of *MHD* dataset sliced along each axis.

C.10 Turbulence Gravity Cooling (TGC)

The TGC dataset (Ohana et al., 2024), simulates a turbulent fluid with gravity. We provide in fig. 14 a trajectory sample. Additional details can be found in (Ohana et al., 2024).



(a) Slice along x. (b) Slice along y. (c) Slice along z.

Figure 14: Trajectory sample of *TGC* dataset sliced along each axis.

D Architecture details

We provide additional architecture details about ECHO's auto encoder in section D.2 and generative model section 3.1.

D.1 Inference model

We describe here several instance of the inference process corresponding to different tasks, that can be handled by ECHO. This is an illustration of the inference formalism introduces in section 2.1.

- **Forward prediction:** $\mathcal{O}$ contains the first L frames of a trajectory, encoded as $\boldsymbol{z}^{0:L'-1} = E_\phi(\boldsymbol{u}^{0:L-1}) \in \mathbb{R}^{M\times L'\times d}$. The model predicts the missing tokens $\boldsymbol{z}^{\mathcal{M}'} = \{\boldsymbol{z}^{L'}, \dots, \boldsymbol{z}^{T'}\}$, completing the latent trajectory.
- **Interpolation:** $\mathcal{O}$ consists of L non-consecutive frames $\{\boldsymbol{u}^{t_1}, \dots, \boldsymbol{u}^{t_L}\}$ with $\{t_0, \dots, t_{L-1}\} \subset [0, T]$. Each observed frame is encoded spatially as $\boldsymbol{z}^{t_\ell} = E_\phi(\boldsymbol{u}^{t_\ell}) \in \mathbb{R}^{M\times d}$, yielding a set of latent observations $\boldsymbol{z}^{\mathcal{O}'}$. The generative model then reconstructs the missing temporal dynamics by producing the full latent trajectory $\boldsymbol{z}^{0:T'}$, thus interpolating the unobserved tokens $\boldsymbol{z}^{\mathcal{M}'} = \boldsymbol{z}^{0:T'} \setminus \boldsymbol{z}^{\mathcal{O}'}$.
- **Backward prediction:** $\mathcal{O}$ contains the last frames of a trajectory $\{\boldsymbol{u}^{t_{T-L+1}}, \dots, \boldsymbol{u}^{t_T}\}$, encoded as $\boldsymbol{z}^{\mathcal{O}'}$. The model reconstructs earlier dynamics by generating $\boldsymbol{z}^{0:T'}$, recovering $\boldsymbol{z}^{\mathcal{M}'}$.
- **Initial value problem:** $\mathcal{O}$ contains only $\boldsymbol{u}^0$ (optionally with $\boldsymbol{\gamma}$), encoded as $\boldsymbol{z}^0 \in \mathbb{R}^{M\times 1\times d}$. The model then generates $\boldsymbol{z}^{1:T'} = \boldsymbol{z}^{\mathcal{M}'}$, yielding the full trajectory $\boldsymbol{z}^{0:T'}$.
- **Conditional / unconditional generation:** $\mathcal{O} = \varnothing$. The model initializes $\boldsymbol{z}_0^{\mathcal{M}'} \sim \mathcal{N}(0, I) \in \mathbb{R}^{M\times T'\times d}$ and generates $\boldsymbol{z}^{0:T'}$, either unconditionally or conditioned on PDE parameters $\boldsymbol{\gamma}$.

Spatial tasks ECHO's encoder–decoder architecture naturally supports a variety of spatial tasks. Thanks to the continuous convolution layers, the model can encode an arbitrary number of input points and decode at any desired spatial location. As a result, ECHO can perform spatial interpolation, super-resolution, and related tasks simply by querying the decoder at the appropriate target points.

- **Spatial interpolation**: Ξ denotes an incomplete irregular grid. During decoding, we query the decoder at the missing locations, effectively interpolating the field on the unobserved points.
- **Super-resolution**: During decoding, we query a finer grid that includes intermediate locations between the original points, thereby interpolating the input data at a higher spatial resolution.

We provide examples of super-resolution in table 4 and spatial interpolation (reconstruction from irregular partial grids) in tables 3 and 22.

D.2 Encoder-Decoder

The encoder–decoder architecture consists of two stages. First, a continuous convolution layer regularizes the (potentially) irregular input grid (section D.2.1). Second, a spatio-temporal CNN compresses the regularized representation to the desired token dimension (section D.2.2), as illustrated in fig. 15.

D.2.1 Interpolation with continuous convolution

When encoding (possibly) irregular high-dimensional data, the computational burden falls on the encoder, which must process a large number of spatial points. The first layer plays a crucial role in reconstruction performance: applying a very high compression ratio leads to information loss, whereas avoiding compression results in significant computational cost (as detailed in section 4). In ECHO, we leverage continuous convolutions (Hagnberger et al., 2025) to handle irregular grids.

We now detail the continuous convolution layers used in the encoder and decoder.

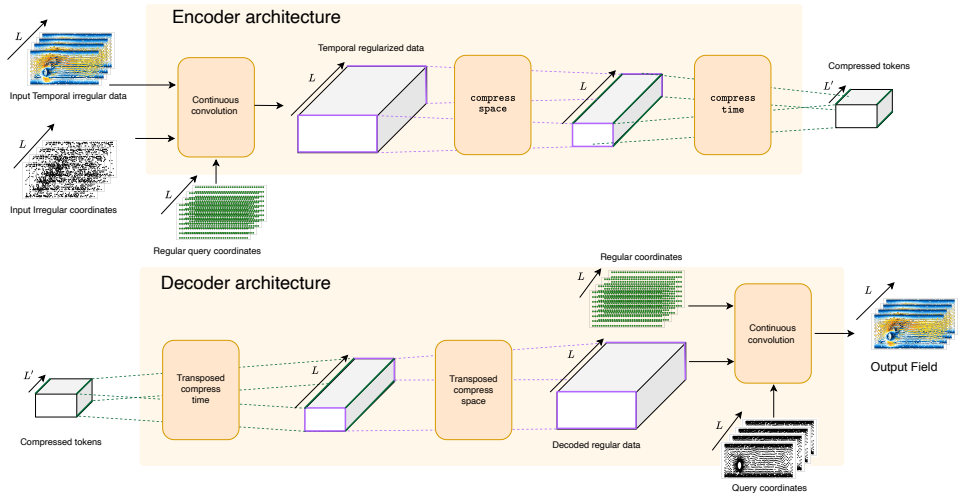


Figure 15: Encoder–decoder architecture of ECHO. The auto-encoder uses a continuous convolution layer to process irregular input grids with an arbitrary number of points, mapping the dynamics onto a regular latent grid. This latent representation is then compressed by a hierarchical CNN encoder. The decoder mirrors this process with a hierarchical CNN decoder followed by a continuous convolution layer, enabling queries at any desired output location.

Continuous convolution for the encoder. In the encoder, the inputs are: (i) a physical field with C_{in} channels, $f^{\mathcal{O}}(\mathcal{X})$, defined on coordinates $\mathcal{X} = \{x_p\}_{p=1}^{|\mathcal{X}|}$ (with $|\mathcal{X}|$ the number of spatial points), and (ii) a regular latent grid $\Xi = \{\xi_j\}_{j=1}^{S}$, where S is the number of latent grid points.

With these notations, the continuous convolution at an output location $\xi_j \in \Xi$ is given by

$$(f^{\mathcal{O}} * k)_o(\xi_j) = \sum_{i=1}^{C_{\text{in}}} \sum_{p \in \text{RF}(\xi_j)} f_i(x_p)\, k_{i,o}(\xi_j - x_p), \tag{28}$$

where $\text{RF}(\xi_j)$ denotes the receptive field around ξ_j, and $k_{i,o}$ is a kernel parameterized by a neural network. Each output on the regular latent grid is thus obtained by convolving the input field with this learned kernel. Importantly, there is no constraint on the number of input points $|\mathcal{X}|$, so the encoder can handle arbitrary point clouds.

Continuous convolution for the decoder. The final decoder layer has a similar form. Let f^{DEC} be the decoded latent representation with d channels, defined on the regular grid Ξ (the same latent grid as in the encoder). Given a set of query coordinates $\mathcal{X} = \{x_i\}$ in physical space, the reconstructed field at a query point x_i is computed as

$$(f^{\text{DEC}} * k)_o(x_i) = \sum_{c=1}^{d} \sum_{p \in \text{RF}(x_i)} f_c^{\text{DEC}}(\xi_p)\, k_{c,o}(x_i - \xi_p), \tag{29}$$

where $k_{c,o}$ is again a learned kernel and $\text{RF}(x_i)$ denotes the receptive field around the query location x_i.

Because the convolution kernel k is learned during training and the formulation is defined for arbitrary query coordinates, the model can be evaluated at any location x_i. This enables a range of spatial tasks such as spatial interpolation and super-resolution, as demonstrated in sections 4 and F.

ECHO's architectural modifications In order to improve the scalability of this layer, we perform some architectural modifications.

- **Fixed latent grid**: The latent grid is fixed to a dense regular grid (up to 36k points for *Vorticity*). This leads to a lower memory consumption at training. This is also motivated by

the use of CNN-based compression layers that needs to operate on regular grids and enables efficient and scalable encoding.

- **Chunk**: We perform chunking for very dense grids, i.e. we compute the output locally by iterative on smaller amount of query points.

D.2.2 Spatial and Temporal compression

For the architecture of the CNN, we use a MagViT architecture (see for example Yu et al. (2023); Koupaï et al. (2025) for additional details). In our case, the encoder CNN takes as input the output of the interpolation module detailed above, ie a regular representation of the input field. The regularized physical space tensor is then compressed through a stack of three building blocks: `residual`, `compress_space`, and `compress_time`. Finally, a last layer projects the representation back to the target token dimension. The decoder mirrors this compression pipeline. This type of architecture has been shown to be effective for spatio-temporal compression (Serrano et al., 2025; Yu et al., 2023; Koupaï et al., 2025). The three types of layers used in the compression module are detailed below (see (Koupaï et al., 2025)).

`residual` **blocks:** The `residual` block processes the input while preserving its original shape. It consists of a causal convolution with kernel size k, followed by a linear layer and a Global Context layer adapted from Cao et al. (2023). If the output dimensionality differs from the input's, an additional convolution is used to project the spatial channels to the desired size.

`compress_space` **blocks:** The `compress_space` block reduces spatial resolution by a factor of 2^d, i.e., each spatial dimension is downsampled by a factor of 2 using a convolutional layer with stride $s = 2$. The kernel size k and padding p are set accordingly, with $p = k//2$. To ensure that only spatial dimensions are compressed, inputs are reshaped so that this operation does not affect the temporal axis.

`compress_time` **blocks:** The `compress_time` block performs temporal compression similarly to the spatial case, but operates along the time dimension. To preserve causality, padding is applied only to the past, with size $p = k - 1$, so that a frame at time t attends only to frames at times $< t$. A convolution with stride $s = 2$ is used to reduce the temporal resolution by a factor of 2.

ECHO's architectural choices In our experiments, we compress physical fields by separating spatial and temporal compression. We start by stacking several `compress_space` blocks to reduce the spatial dimensions of the tokens. Then, we process it using a `residual` layer. Since these layers, make use of 3D causal convolution, they allow to process jointly the spatial and temporal dimensions. Finally, we compress the temporal dimensions with `compress_time` layers. This gives us an encoded representation of the input physics. The detailed number of layers and hyper-parameters related to the encoder/decoder modules are shown in table 12.

E Implementation details

The code has been written in Pytorch (Paszke et al., 2019). All experiments were conducted on a H100. We estimate the total compute budget—including development and evaluation—to be approximately 1000 GPU-days.

E.1 Evaluation protocol of the process

E.1.1 Baseline details

We detail the baseline architectures used to evaluate ECHO in both forward and backward prediction, corresponding to experiments in section 4.1. All baselines follow a comparable training protocol (see section E.1.2). For forward and backward experiments, we assume 4 known timesteps and autoregressively predict the remainder, except for ECHO (and its deterministic variant), which generates the full trajectory in a single forward pass.

FNO. For the Fourier Neural Operator (FNO) (Li et al., 2021), we followed the authors' recommendations and concatenated temporal history directly to the input channels. We used 10 Fourier modes in both 1D and 2D, a channel width of 128, and stacked 4 spectral layers.

BCAT. BCAT (Liu et al., 2025) is a deterministic block-wise causal transformer for spatio-temporal dynamics, originally designed for multi-physics problems. We adapt it to parametric PDEs. BCAT performs autoregression in physical space and uses spatial patches to reduce token count, similar to Vision Transformers (Dosovitskiy et al., 2021).

Table 7: BCAT hyperparameters used across all datasets.

Hyperparameter	Value
Patch size	8
Transformer depth	6
Hidden size	512
MLP ratio	2
Number of heads	8
QK normalization	True
Normalization type	RMS
Activation	SwiGLU
Positional embedding	Sinusoidal

AViT. The Axial Vision Transformer (AViT) (McCabe et al., 2024) applies attention separately along spatial and temporal dimensions. We use the same configuration as BCAT (Table 7), which we found to perform best on our PDE datasets.

Transolver++. Transolver++ (Luo et al., 2025) extends (Tran et al., 2023) for dense input grid, with improved parameterization and efficiency. We followed the recommended setup summarized below, provided in the reference paper.

ENMA. For ENMA (Koupaï et al., 2025), we implemented the generation architecture given in the original paper. We followed the authors recommenation for the hyper-parameters. Its configuration across datasets is summarized in table 9.

E.1.2 Training details

All models were trained under the same protocol for fair comparison. Unless specified, we used the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, a cosine learning rate schedule from 10^{-3} to 10^{-5}, and linear warm-up over the first 2000 steps. Baselines were trained with the same schedule and selection criterion as ECHO, using the best checkpoint on train error.

Table 8: Transolver++ hyperparameters used across all datasets.

Hyperparameter	Value
Number of layers	8
Hidden size	128
Attention heads	8
MLP ratio	4
Dropout	0
Physics slices	32
Activation	GELU

Table 9: ENMA generation hyperparameters.

Hyperparameter	Active Matter	Rayleigh–Bénard	Gray–Scott	ASM
VAE embedding dim	4	4	4	8
Number of tokens	16	16	64	256
Patch size	2	2	1	2
Spatial Transformer depth	8	8	8	8
Causal Transformer depth	8	8	8	8
Hidden size	512	512	512	512
MLP ratio	2	2	2	2
Attention heads	8	8	8	8
Dropout	0	0	0	0
QK normalization	True	True	True	True
Normalization type	RMS	RMS	RMS	RMS
Activation	SwiGLU	SwiGLU	SwiGLU	SwiGLU
Positional embedding	Sinusoidal	Sinusoidal	Sinusoidal	Sinusoidal
FM steps	10	10	10	10

E.1.3 Fairness of Baseline Comparisons and Scaling Limits.

To ensure a rigorous and fair comparison, we carefully controlled for parameter counts, memory tuning, and training budgets across all baseline models. For memory-intensive baselines, we carefully searched for the best tradeoff between memory consumption and performance, to fit them within the 80GB memory limit of our H100 GPUs. Despite these tuning efforts, models relying on dense spatial attention (e.g., AROMA) or direct physical-space processing inherently scale poorly with grid resolution. The out-of-memory (OOM) failures reported for baselines on table 4 are therefore not due to a lack of optimization, but rather fundamental architectural bottlenecks that ECHO effectively circumvents via its joint spatio-temporal compression. Additionally, this experiment corresponds to a super-resolution task on the Vorticity dataset, because the grid size considered during training was limited to 512×512.

On the other hand, fig. 4 shows a separate experiment. The setting is the following: we instantiate models with a fixed latent size, except for Transolver ++, for which we considered the setting of the

Table 10: Training hyperparameters used across datasets.

Hyperparameter	Value
Epochs	200
Batch size	32
Learning rate	10^{-3} (cosine decay)
Weight decay	10^{-4}
Grad clip norm	1
Betas (β_1, β_2)	(0.9, 0.95)

authors (Luo et al., 2025). Then we generate random grid, with growing size, that we fed into the networks (input and query size are similar). The objective here is to find the limits of the models in terms of memory consumption, rather than evaluation their capacities on the super-resolution task.

Model sizes. Table 11 reports the number of parameters of the models in table 1. ECHO is smaller than FNO, AViT and ENMA, so its accuracy cannot be attributed to a larger parameter count. For Transolver++, larger configurations exceeded GPU memory.

Table 11: Number of parameters of the models compared in table 1.

Dataset	Transolver++	FNO	AViT	BCAT	ENMA	ECHO
Rayleigh–Bénard	5.5M	75M	51M	27M	72M	48M
Active Matter	5.5M	75M	53M	28M	72M	48M

Cause of the out-of-memory failures. At moderate resolution (table 3), the cost of encoding the raw mesh is paid by shrinking the models to fit the memory budget. Attention-based encoders must use a smaller latent size on *Vorticity* and *Shallow-Water* and compress the mesh in a single stage. The first layer of CALM-PDE is more memory-intensive than ours, so it admits fewer latent points and therefore fewer hierarchical blocks. At higher resolution (table 4), this cost can no longer be paid and every baseline runs out of memory, including those without attention (GINO, CORAL, CALM-PDE).

E.2 Encoder-Decoder implementation protocol

E.2.1 ECHO encoder-decoder's Architecture hyper-parameters

We present in table 12 the hyper-parameters used for the experiments done in section 4.3 for our encoder-decoder module.

Table 12: Hyper-parameters of the encoder and decoder components of ECHO.

Parameter	Vorticity	Shallow-Water	Eagle	Cylinder Flow
Token dimension	32	32	32	32
Interpolation module				
Regularized grid	192×192	$64 \times 128 \times 1$	64×32	64×16
Hidden dimension	64	64	64	64
Receptive field	0.01	0.01	0.01	0.01
Softmax temperature	1	1	1	1
Chunk size	4096	4096	4096	4096
Max neighbors	512	512	512	16
Latent grid type	Euclidean	Spherical	Euclidean	Euclidean
Spatial dimension	2	2	3	2
Compression module				
Spatial compression layers	3	2	2	2
Temporal compression layers	1	2	2	2
Residual layers	One between each compression layer			
Compression kernel size	3	3	3	3
Causal input layer kernel size	3	3	3	3
Causal output layer kernel size	3	3	3	3

E.2.2 Baseline details

We detail here the hyper-parameters used for the baselines considered in section 4.3. If not explicitly stated, we follow the authors recommendation in the respective papers.

In table 14, we report the shapes of the trajectory in the physical space and in the latent space for each baseline. GINO, CORAL, AROMA, ENMA and CALM-PDE only compress information spatially. In ECHO, both temporal and spatial space can be compressed.

Matching latent size rather than parameter count. Encoder–decoders are matched on compression rate, that is, on the size of the latent representation, rather than on parameter count. For each baseline, we follow the authors' recommendations and target a comparable latent size, so that all methods compress the input to approximately the same information budget; for instance, on *Vorticity*, GINO uses a latent representation of $10 \times 576 \times 32 = 184{,}320$ values. The exceptions are ECHO, whose temporal compression allows a smaller latent size, and AROMA, whose latent size is constrained by memory. For hierarchical methods such as CALM-PDE, we stack as many compression blocks as needed to reach this size. For Perceiver-style encoders such as AROMA and ENMA, enlarging the encoder beyond this point has limited effect, since the single cross-attention projection has already discarded much of the input signal. Section F.8.1 provides the controlled version of this observation: at a fixed final latent size, adding intermediate compression stages improves reconstruction, an effect that parameter count does not capture.

Table 13 reports the resulting encoder sizes. AROMA, ENMA and CALM-PDE do not fit in GPU memory in their published configurations (colored cells in table 3); the counts correspond to the reduced configurations used in our experiments. Model size plays a minor role in scalability: AROMA and CALM-PDE use encoders two orders of magnitude smaller than that of ECHO and still exceed memory on the 1024×1024 grid, because their cost grows with the number of input points (fig. 4). All methods use a batch size of 1 at inference and the same temporal window.

Table 13: Number of parameters of the encoder–decoders compared in table 3.

Dataset	GINO	CORAL	AROMA	ENMA	CALM-PDE	ECHO
Vorticity	18M	23M	0.5M	1.9M	0.5M	33M
Shallow-Water	19M	21M	0.5M	5M	0.2M	30M

Table 14: Physical and latent space shapes in the experiments of table 3, in the format temporal size $\times$ spatial size $\times$ token dimension. The first row gives the shape of the trajectories. On datasets with many input points, some baselines use a smaller latent shape (in red) to fit in GPU memory.

Model	Vorticity	Shallow-Water	Eagle	Cylinder Flow
Trajectory	$10 \times (512 \times 512) \times 1$	$10 \times (128 \times 256) \times 2$	$200 \times 3000 \times 3$	$60 \times 2000 \times 3$
GINO	$10 \times 576 \times 32$	$40 \times 512 \times 32$	$100 \times 512 \times 32$	$60 \times 64 \times 32$
CORAL	10×18432	40×16384	200×4096	50×2048
AROMA	$10 \times 64 \times 32$	$40 \times 128 \times 32$	$200 \times 128 \times 32$	$60 \times 64 \times 32$
CALM-PDE	$10 \times 576 \times 32$	$40 \times 512 \times 32$	$200 \times 128 \times 32$	$60 \times 64 \times 32$
ECHO	$5 \times 24 \times 24 \times 32$	$10 \times 16 \times 8 \times 32$	$50 \times 16 \times 8 \times 32$	$15 \times 16 \times 4 \times 32$

GINO In GINO, a graph operator is proposed relying on a GNO block to encode mesh points into a latent space. The resulting latents are then processed by 8 FNO layers to increase model expressivity. We use the default hyperparameters provided in the `neuralop` package (Kossaifi et al., 2025; Kovachki et al., 2023).

CORAL CORAL is a operator learning framework relying on cordinate-based networks for solving PDEs on general geometries. The INRs are learned via a second-order CAVIA-style meta-learning scheme that optimizes shared parameters so that, for each new function, only a few inner-loop gradient steps are required to obtain an accurate per-sample latent code from sparse observations. We use 3 inner-loop steps with an inner learning rate of 10^{-2}. The INR has 6 layers of width 256, and the hypernetwork has 1 hidden layer with width 256. We refer to (Serrano et al., 2023) for additional details.

AROMA The AROMA baseline uses cross-attention to directly compress trajectories into a latent space. We follow the configuration proposed in (Serrano et al., 2024) and set the number of latent tokens as detailed in table 14. Due to GPU memory limitations, we use fewer tokens than the other baselines on two datasets (Vorticity and Shallow-Water).

CALM-PDE For CALM-PDE, we follow the configuration from (Hagnberger et al., 2025), using 2 or 3 hierarchical convolutional layers where the spatial resolution is gradually reduced while the number of channels increases. For the first layer, which processes irregular grids, we select the largest latent grid that fits in GPU memory, and then apply a sequence of $\times 2$ downsampling steps until we reach the desired token resolution. As reported in table 3, GPU memory constraints forced us to use a smaller first-layer latent grid than ECHO on three datasets: 2048 for Vorticity, 1024 for Shallow-Water, and 512 for Eagle.

E.2.3 Training details

Reconstruction performances To ensure fairness, we trained all encoder–decoders using a unified procedure. Specifically, we optimized a relative L^2 loss between the auto-encoder output and the original function evaluated on the full spatial grid (the 100% setting).

Models are trained with the AdamW optimizer, using an initial learning rate of 10^{-3} that is annealed to 10^{-7} with a cosine schedule. Each model is trained for 20 hours, and the batch size is tuned to best utilize the available GPU memory on an NVIDIA H100 (80,GB), with the chosen values reported in table 15.

For ECHO, as we do spatio-temporal encoding, we divide our training into two stages, as explained in section 3.2. We report the hyper-parameters considered in table 16.

Table 15: Training batch sizes used across datasets, for reconstruction (*Rec.*) and time-stepping (*TS*).

Model	Task	Vorticity	Shallow-Water	Eagle	Cylinder Flow
GINO	Rec.	8	16	32	32
	TS	4	16	8	16
CORAL	Rec.	1	2	16	32
	TS	1	4	16	16
AROMA	Rec.	1	1	4	32
	TS	1	1	4	16
ENMA	Rec.	1	1	1	8
	TS	1	2	4	16
CALM-PDE	Rec.	1	1	2	2
	TS	8	16	8	16

Table 16: Training hyperparameters used across datasets for the low- and high-resolution stages.

Hyperparameter	Low resolution	High resolution
Epochs	200	10
Batch size	8	8
Learning rate	10^{-3} (cosine decay)	10^{-4} (cosine decay)
Weight decay	10^{-4}	10^{-4}
Grad clip norm	1	1
Betas (β_1, β_2)	(0.9, 0.95)	(0.9, 0.95)

Generative process training All models are evaluated on the forward task, where 4 time-steps are considered as context go generate the trajectory. Hyper-parameters are ported in table 17 across all baselines. For this training, we considered a maximum budget of 20 hours.

Table 17: Training hyperparameters used across datasets for the generative stage.

Hyperparameter	Generative stage
Epochs	200
Batch size	32
Learning rate	10^{-3} (cosine decay)
Weight decay	10^{-4}
Grad clip norm	1
Betas (β_1, β_2)	(0.9, 0.95)

F Additional experiments

To further assess ECHO's capabilities beyond the main-text experiments, we report a series of additional studies covering long-range extrapolation, unconditional generation, robustness to sparse and irregular observations, and key architectural design choices. Specifically, we consider:

- **Out-of-distribution evaluation** on the Vorticity equation: we evaluate the OOD capacities of ECHO on new unseen physical parameters (section F.1).
- **Long-range prediction** on Shallow-Water, where ECHO is evaluated up to $4\times$ its training horizon (section F.2). We also provide a spectra analysis on the Vorticity equation;
- **Generative trajectory sampling** in an unconditional setting on Shallow-Water, including Fréchet Physics Distance (FPD) and efficiency metrics (section F.3);
- **Probabilistic evaluation** of ECHO with 16-member ensembles (CRPS, calibration error, spread-skill ratio) on forward and backward prediction (section F.4);
- **Initial value problem (IVP) evaluation**, where ECHO generates full trajectories from a single initial frame and PDE parameters (section F.5);
- **Temporal interpolation tasks**, in which ECHO reconstructs long missing segments between observed frames across four datasets (section F.6);
- **Encoding sparse data / spatial interpolation**, where only 50% or 10% of the input grid is observed and the full grid must be reconstructed, with 10% constituting a strong OOD sparsity level (section F.7).

In addition, we provide a set of ablation studies in section F.8, analyzing:

- the impact of *hierarchical compression depth* in the encoder on reconstruction accuracy (section F.8.1);
- the contribution of the *refinement stage* for full-resolution Vorticity reconstruction (section F.8.2);
- the robustness of the *flow-matching generative process* to the choice of ODE solver and number of integration steps at sampling time (section F.8.3).

F.1 Out-of-distribution evaluation

In this section, we probe ECHO's ability to (i) generalize to out-of-distribution PDE parameters and (ii) specialize at a specific task (Forward generation here) on new data. We vary the viscosity parameter: training uses $\nu \in [10^{-3}, 10^{-4}]$, while OOD evaluation uses $\nu \in [10^{-4}, 10^{-5}]$, yielding more turbulent, higher-Reynolds flows with different initial conditions. We consider two settings: (i) a strict **zero-shot** scenario, where the pre-trained model is directly applied to the new regime, and (ii) a **few-shot adaptation** scenario, where the model is lightly fine-tuned on a small set of OOD trajectories for the forward prediction task.

Table 18: OOD generalization on Vorticity: performance of ECHO under distribution shift, with zero-shot and few-shot adaptation. Relative L^2 error (lower is better).

Setting	Rel. L^2
In-distribution	1.17e-1
OOD zero-shot	3.39e-1
OOD few-shot	1.12e-1

Results As shown in Table 18, ECHO displays strong zero-shot transfer to the more turbulent regime. Modest few-shot fine-tuning on OOD forward trajectories further improves accuracy, illustrating how a multi-task pre-trained operator can be efficiently adapted to new dynamics, paving the way toward universal PDE surrogates that are trained once and quickly specialized at low cost.

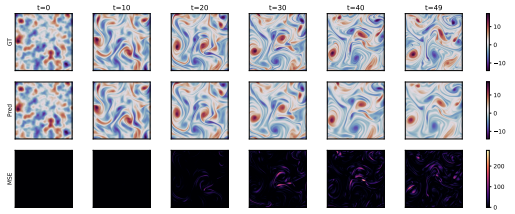
F.2 Long-range prediction

F.2.1 Long-range prediction on Vorticity

We further illustrate ECHO's ability to generate long-range predictions beyond the training horizon on the Vorticity dataset, which exhibits turbulent regimes. Figure 16 illustrates that ECHO remains consistent in long-rollouts, even on turbulent dynamics such as the Vorticity PDE. Note that we observed in fig. 1 (middle) that ECHO accumulates less error than baselines.

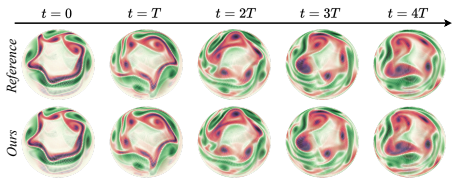
F.2.2 Long-range prediction on Shallow-Water

To further assess the ability of ECHO to perform long-range prediction, we extend the experiments of section F.2.1 to the Shallow-Water equation. On this dataset, ECHO is trained on trajectories of length $T = 40$ time steps, and evaluated in an out-of-distribution regime up to $4T = 160$.



Horizon	$0.5T$	T	$1.5T$	$2T$	$2.5T$
Rel. L^2	7.70e-2	1.72e-1	2.62e-1	3.79e-1	4.79e-1

Figure 16: Vorticity: long-range prediction with ECHO. Relative L^2 error at increasing horizons.



Horizon	T	$2T$	$3T$	$4T$
Rel. L^2	1.21e-2	2.02e-2	2.11e-2	3.15e-2

Figure 17: Shallow-Water: long-range prediction with ECHO ($T = 40$, evaluated up to $4T$). Relative L^2 error at increasing horizons.

Results Figure 17 shows that the error increases smoothly with the rollout horizon, from 1.21×10^{-2} at T to 3.15×10^{-2} at $4T$. This moderate and monotonic degradation indicates that ECHO produces stable long-range forecasts even when extrapolating four times beyond its training horizon.

F.2.3 Spectral analysis on the Vorticity generation

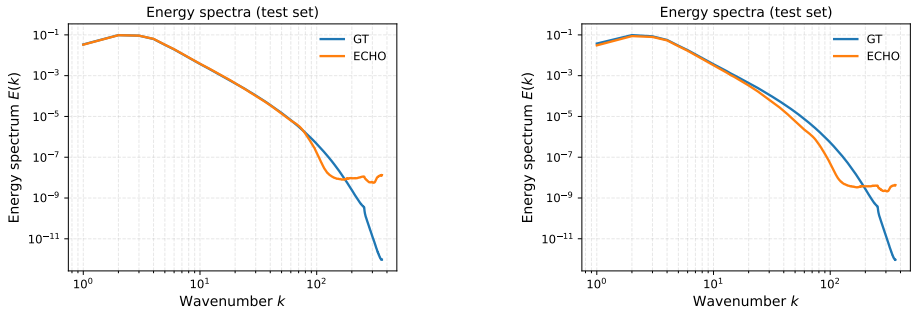
While relative L2 is a standard metric in the PDE literature, it can hide over-smoothing, especially on turbulent fields where high-frequency features are important.

To better assess ECHO's ability to preserve small-scale structures, we perform a spectral analysis on the *Vorticity* dataset, both within the in-distribution training horizon and far beyond it. This type of analysis is standard in fluid dynamics and characterizes how energy is distributed across spatial scales, from large eddies at low wavenumbers k to fine-scale structures at high k.

We proceed as follows: (i) we transform each physical field $u(x)$ into the frequency domain via FFT, obtaining Fourier coefficients $\mathcal{F}(u)(k)$ at accessible wavenumbers k; (ii) we compute the corresponding energy spectrum $E(k) = \frac{1}{2}|\mathcal{F}(u)(k)|^2$; (iii) we average the energy over all wavevectors with the same magnitude $|k|$ to obtain an isotropic spectrum.

The resulting spectra are shown in fig. 18 (done for the forward task): generation within the training horizon (up to $T = 20$, left) and long-range generation beyond the training horizon (up to $T = 50$, right). Each curve reports the energy spectrum averaged over all samples and time-steps in the corresponding temporal range.

Results ECHO faithfully reconstructs the physics of the system across the most energy-dominant scales. The only discrepancy is a slight overestimation of energy at the very finest scales (high spatial frequencies), where most neural surrogates are known to fail. For a simulation running beyond its training horizon, ECHO shows good stability (the large scales are accurate). Beyond the training distribution, ECHO succeeds to preserve the low and medium scales, but fail to capture high and very-high spatial frequencies.



(a) Spectral analysis for generation of a trajectory inside the training horizon ($T = 20$).

(b) Spectral analysis for generation of a trajectory beyond the training horizon ($T = 50$).

Figure 18: Spectral analysis on a trajectory generation with ECHO on the *Vorticity* dataset.

Origin of the high-frequency discrepancy. The deviation at high wavenumbers has two sources. The auto-encoder is trained with a relative L^2 reconstruction loss, which is dominated by the energy-carrying large scales and therefore fits low frequencies first, so the finest scales are the first affected by compression. Within the training horizon, this appears as a small excess of energy at the highest wavenumbers (fig. 18a). Over long rollouts, small errors at fine scales propagate to intermediate scales, so the departure from the reference moves toward lower wavenumbers beyond the training horizon (fig. 18b). Frequency-aware reconstruction objectives are a possible direction to reduce this effect.

F.3 Generative data sampling

As a demonstration of **ECHO**'s generative capability, we assess its ability to produce diverse and physically plausible trajectories (on Shallow-Water equation) and compare its efficiency to existing neural surrogates. We evaluate its generative quality using the **Fréchet Physics Distance (FPD)** (Koupaï et al., 2025), an adaptation of the Fréchet Inception Distance widely used in image synthesis. FPD is computed in a compact 64-dimensional feature space extracted by a lightweight CNN encoder trained on physical trajectories, enabling semantically meaningful comparisons while avoiding the curse of dimensionality. Lower FPD indicates that generated trajectories are statistically closer to the training data distribution. For this evaluation, ECHO is trained in a fully *unconditional* setting, where the entire trajectory is noised and then generated.

Beyond fidelity, we report standard **efficiency metrics** relevant for large-scale PDE surrogates:

- **Throughput (Training)** — the number of samples processed per second during training (higher is better);
- **Latency** — average wall-clock time to generate a single trajectory at inference (lower is better);
- **Parameter count** — total learnable parameters, which provides a proxy for model complexity.

Results ECHO achieves the lowest FPD by a wide margin (1.1×10^{-3}), indicating its generated trajectories align extremely closely with the true data distribution, significantly outperforming existing efficient surrogates such as AROMA (1.5×10^{-2}) and CALM-PDE (1.0×10^{-1}). While slightly heavier in parameters than minimalistic encoders, ECHO maintains competitive **throughput** (35 samples/s) and extremely low **inference latency** (0.21s/trajectory), close to the fastest baseline AROMA (0.19s) while being substantially more accurate.

F.4 Probabilistic evaluation

All results in the main text are computed from a single sample per test trajectory, for generative and deterministic models alike. Single-sample errors do not assess the predictive distribution of a generative model. We therefore sample 16 trajectories per test case and report three standard

Table 19: Efficiency and generative fidelity on Shallow-Water. Every encoder–decoder is combined with our generative model. Throughput and latency are measured on identical hardware; FPD measures the distributional alignment with ground-truth trajectories. Best results in **bold**, second best underlined.

Encoder–decoder	#Params	Training throughput (samples/s) ↑	Latency (s) ↓	FPD ↓
GINO	19M	55.7	1.83	1.68e-1
CORAL	21M	4.3	0.75	1.31e-1
AROMA	0.5M	32.8	**0.19**	1.55e-2
CALM-PDE	2M	**60.5**	0.24	1.02e-1
ECHO	30M	35.0	0.21	**1.12e-3**

probabilistic metrics, together with the relative L^2 error of the ensemble mean. The **CRPS**, in its fair ensemble form, is a proper scoring rule that evaluates the full predictive distribution. The **RMSCE** (root mean squared calibration error) measures the gap between nominal and empirical coverage of the ensemble quantiles. The **SSR** (spread-skill ratio, with finite-ensemble correction) compares the ensemble spread with the error of the ensemble mean: a value of 1 indicates that the spread matches the realized error, and values below 1 indicate over-confident ensembles.

Table 20: Probabilistic evaluation with 16-member ensembles, using the same windows and number of generated frames as table 1. Rel. L^2 is computed on the ensemble mean. Bold marks the best forward result per dataset; ECHO backward rows use identical settings.

Dataset	Task	Model	CRPS ↓	RMSCE ↓	SSR → 1	Rel. L^2 ↓
Gray–Scott	Forward	ENMA	8.68e-3	3.16e-1	0.43	3.96e-2
		ECHO	**8.66e-3**	**9.34e-2**	**0.93**	**3.90e-2**
	Backward	ECHO	9.69e-3	7.94e-2	1.06	4.41e-2
Rayleigh–Bénard	Forward	ENMA	**9.47e-3**	2.03e-1	0.53	9.38e-2
		ECHO	1.04e-2	**8.97e-2**	**0.86**	**9.02e-2**
	Backward	ECHO	1.30e-2	4.77e-2	0.90	1.14e-1
Active Matter	Forward	ENMA	1.11e-1	4.51e-1	0.12	2.98e-1
		ECHO	**6.79e-2**	**1.73e-1**	**0.67**	**2.33e-1**
	Backward	ECHO	6.85e-2	1.19e-1	0.80	2.41e-1
ASM	Forward	ENMA	5.16e-3	1.97e-1	0.59	3.41e-1
		ECHO	**1.65e-3**	**1.37e-1**	**0.82**	**1.19e-1**
	Backward	ECHO	1.77e-3	1.27e-1	0.96	1.20e-1

Forward prediction. On forward prediction (table 20), ECHO is well calibrated, with RMSCE between 9.0×10^{-2} and 1.7×10^{-1} and SSR between 0.67 and 0.93. ENMA is under-dispersed on every system, with SSR between 0.12 and 0.53, meaning that its ensembles are over-confident relative to their realized error. ENMA obtains a slightly lower CRPS on *Rayleigh-Bénard*; on the three other systems, ECHO is better on all four metrics.

Backward prediction. Backward prediction can be ill-posed, since several earlier states may be consistent with the observed frames, which makes the calibration of the ensembles the relevant property. Backward performance of ECHO is close to its forward performance on all four systems, with CRPS increasing by 1% (*Active Matter*) to 25% (*Rayleigh-Bénard*). The backward ensembles are well calibrated, with RMSCE between 4.8×10^{-2} and 1.3×10^{-1} and SSR between 0.80 and 1.06: the predicted spread reflects the actual uncertainty of the reconstruction.

F.5 Initial Value Problem

We further evaluate ECHO in an *initial value problem* (IVP) setting, where the model uses as context only the first frame of the trajectory and the governing PDE parameter (here, the viscosity) as inputs. This setting tests the model's ability to generate a full trajectory from minimal information, a common scenario in real-world PDE simulations.

We consider the **Vorticity** system on a 512×512 grid and report relative L2 reconstruction error over the full rollout. When conditioned only on the initial state and viscosity, ECHO achieves a relative L2 of 1.15×10^{-1}, demonstrating robust trajectory generation despite extremely sparse input.

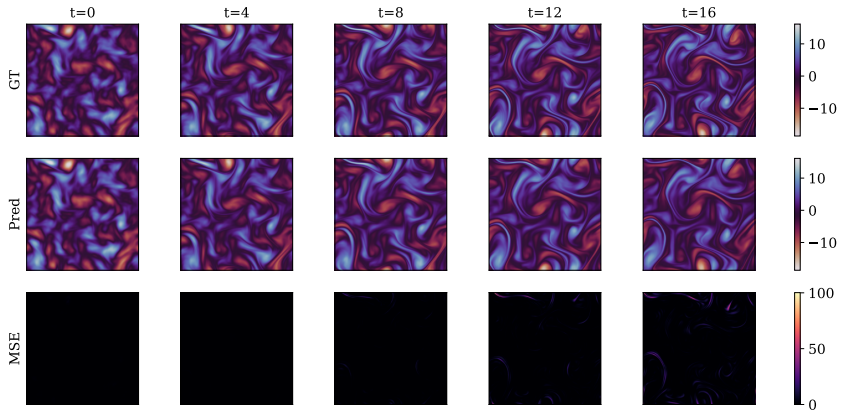


Figure 19: **Initial value problem on Vorticity** (512×512)**.** Ground truth (GT), ECHO predictions, and per-frame error maps when generating the full trajectory from only the first state and PDE viscosity.

F.6 Interpolation tasks

In addition to the forward and backward prediction settings discussed in the main text, we evaluate ECHO on a temporal interpolation task. Given the first 2 and last 2 frames of each trajectory as context, the model must reconstruct all intermediate frames (11 frames for Rayleigh–Bénard and ASM, 18 for Active Matter, and 36 for Gray–Scott). We report the Relative L^2 over the interpolated segment,together with backward and forward errors, in table 21.

Results ECHO shows strong interpolation capabilities across all four systems, both for the deterministic and generative version. These results indicate that ECHO reliably fill in long missing segments between observed states.

F.7 Encoding sparse data - Spatial interpolation

In addition to table 3, we evaluate the encoder–decoder of ECHO and all baselines under more challenging sparse-input regimes. In this experiment, the models receive only a subset of the input grid and must reconstruct the full-resolution output grid. We consider two sampling levels at test time: an intermediate sparsity of 50% of the input points, and an extreme sparsity of 10%.

The 10% setting is out-of-distribution, since all models were trained with input samplings ranging only from 25% to 75%. This evaluation therefore probes the robustness of each method to significantly more aggressive subsampling at inference. The corresponding results are reported in table 22.

Result Overall, table 22 shows that convolution-based hierarchical models (CALM-PDE and ECHO) outperform graph- and INR-based baselines both for reconstruction and time-stepping. At 50% sampling (in-distribution), ECHO achieves the best or second-best performance across most datasets. In the more challenging 10% sampling regime, which constitutes a strong OOD setting relative to the training samplings, all models degrade, but ECHO remains competitive and often matches or surpasses alternative baselines, especially on the more complex Eagle.

Table 21: Backward (*Back.*), forward (*For.*) and temporal interpolation (*Interp.*) results across four dynamical systems. Errors are relative L^2 errors (lower is better). Interpolation is evaluated for ECHO only; latencies are reported in table 1. Best results in **bold**, second best underlined; "–" indicates non-convergence.

	Rayleigh–Bénard			Gray–Scott		
Model	Back.	For.	Interp.	Back.	For.	Interp.
Deterministic models						
FNO	2.47e3	4.23e-1		–	–	
Transolver++	6.34e-1	3.31e-1		4.43e-1	2.34e-1	
BCAT	1.91e-1	1.06e-1		2.19e-1	8.82e-2	
AViT	4.50e-1	1.01e-1		1.66e-1	7.42e-2	
ECHO	2.53e-1	1.32e-1	1.08e-1	8.36e-2	7.66e-2	3.12e-2
Generative models						
ENMA	1.71e-1	9.87e-2		1.08e-1	5.44e-2	
ECHO	**1.16e-1**	**9.28e-2**	**9.02e-2**	**2.53e-2**	**5.12e-2**	**1.78e-2**

	Active Matter			ASM		
Model	Back.	For.	Interp.	Back.	For.	Interp.
Deterministic models						
FNO	–	–		1.87	1.52	
Transolver++	7.33e-1	6.91e-1		1.03	9.64e-1	
BCAT	4.98e-1	4.56e-1		1.95e-1	2.18e-1	
AViT	4.50e-1	4.62e-1		**1.04e-1**	1.52e-1	
ECHO	5.74e-1	3.44e-1	1.47e-1	1.47	2.53e-1	1.88e-1
Generative models						
ENMA	4.27e-1	3.33e-1		1.01	4.12e-1	
ECHO	**3.55e-1**	**2.87e-1**	**1.11e-1**	1.12e-1	**1.32e-1**	**1.00e-1**

Table 22: Encoder–decoder results from sparse inputs: only a fraction $\mathcal{X}_{te}$ of the input grid is observed at test time and the full grid is reconstructed. The 10% setting is out-of-distribution (training samplings range from 25% to 75%). *Rec.* and *For.* denote reconstruction and forward generation; *Irr.* gives the strategy for handling irregular points, and *Hier.* indicates hierarchical (✓) or direct (✗) compression into the latent space. Metrics are relative L^2 errors (lower is better). Best results in **bold**, second best underlined.

			Vorticity		Shallow-Water		Eagle		Cylinder Flow	
Irr.	Encoder	Hier.	Rec.	For.	Rec.	For.	Rec.	For.	Rec.	For.
Intermediate sparsity ($\mathcal{X}_{te} = 50\%$)										
Graph	GINO	✗	9.99e-1	1.00	8.69e-1	1.11	5.80e-1	1.88	7.94e-1	8.65e-1
INR	CORAL	✗	5.14e-1	1.34	2.29e-1	6.88e-1	6.11e-1	1.54	3.04e-1	5.07e-1
Attention	AROMA	✗	5.13e-1	8.43e-1	2.69e-2	4.28e-2	3.09e-1	3.34e-1	**3.17e-2**	2.31e-1
	ENMA	✓	4.39e-1	4.39e-1	7.42e-2	7.53e-2	3.02e-1	3.42e-1	7.77e-2	1.09e-1
Convolution	CALM-PDE	✓	2.76e-1	4.26e-1	2.61e-1	2.84e-1	9.34e-1	9.55e-1	2.50e-1	3.59e-1
	ECHO	✓	**7.32e-2**	**2.23e-1**	**1.94e-2**	**2.00e-2**	**1.88e-1**	**2.90e-1**	4.72e-2	**9.63e-2**
Extreme sparsity, out-of-distribution ($\mathcal{X}_{te} = 10\%$)										
Graph	GINO	✗	9.99e-1	1.00	8.83e-1	1.11	7.82e-1	1.15	7.98e-1	8.64e-1
INR	CORAL	✗	5.27e-1	1.33	2.31e-1	6.86e-1	6.66e-1	1.53	3.77e-1	5.19e-1
Attention	AROMA	✗	5.16e-1	8.44e-1	**4.89e-2**	5.43e-1	5.26e-1	1.01	**7.12e-2**	2.36e-1
	ENMA	✓	4.45e-1	7.78e-1	9.35e-2	8.93e-2	4.72e-1	**6.86e-1**	1.10e-1	1.44e-1
Convolution	CALM-PDE	✓	**3.24e-1**	**4.59e-1**	3.03e-1	3.17e-1	9.79e-1	1.02	3.08e-1	4.14e-1
	ECHO	✓	5.60e-1	9.86e-1	1.06e-1	**8.69e-2**	**3.82e-1**	7.31e-1	1.13e-1	**1.41e-1**

F.8 Ablation studies

F.8.1 Impact of hierarchical compresion

To assess the benefit of hierarchical compression in the encoder, we conduct an ablation study where the final latent space size (the bottleneck resolution) is kept fixed, but the way we reach this compressed space is varied.

A hierarchical level refers to how many successive downsampling stages are applied in space; each level reduces the spatial resolution by a factor of two along each dimension. For instance, a 3-level hierarchy means three consecutive $2\times$ spatial reductions before reaching the final latent grid, whereas level 0 means a single direct projection from the input resolution to the final latent size (no intermediate stages). Importantly, in this experiment we only compress space (not time), so the temporal dimension remains unchanged (table 23).

Table 23: Impact of hierarchical depth in the encoder on reconstruction (relative L^2 error). A higher level means more progressive spatial downsampling, with the same final latent size.

Hierarchical level	0	1	2	3
Rel. L^2	3.94e-1	3.64e-1	1.09e-1	**2.87e-2**

Results Table 23 shows that increasing hierarchical depth dramatically improves reconstruction accuracy while keeping the same final latent size. Without hierarchy (level 0), Relative L^2 is high (3.94e-1), while introducing even one level of progressive compression significantly reduces error. The best performance is achieved with three levels (Relative L^2 2.87e-2), highlighting that iterative, deep compression yields more expressive and informative latent representations than a single aggressive projection.

This validates our design choice of a deep hierarchical encoder: even at fixed compression ratio, progressive downsampling better preserves spatial structure and leads to markedly improved fidelity.

F.8.2 Impact of the refinement stage

To evaluate the contribution of the *refinement stage*, we conduct an ablation on the **Vorticity** dataset at full spatial resolution. Our training strategy first learns from spatio-temporal data that is *spatially subsampled* to reduce memory cost. After this pretraining, we apply a refinement stage that re-encodes each frame individually at *full spatial density*, improving the latent representation for high-resolution inputs.

Table 24: Effect of the refinement stage on full-grid Vorticity reconstruction (relative L^2 error, lower is better).

	w/o refinement	w/ refinement
Rel. L^2	2.24e-1	**6.88e-2**

Results This additional step proves particularly beneficial for highly detailed fields such as Vorticity, where local fine-scale structures are hard to preserve under strong compression. As shown in table 24, adding the refinement stage significantly improves reconstruction accuracy.

F.8.3 Ablations on the generative process

We conduct ablations on the vorticity dataset to validate the robustness of our flow-matching generative design. In particular, we vary both the ODE solver used at sampling time and the number of denoising steps, while keeping the learned model fixed.

Results Overall, the reconstruction errors remain in a very narrow range across all configurations, with differences on the order of 10^{-3}. Simple solvers such as Euler already perform on par with higher-order methods (RK4, DOPRI5), and increasing the number of integration steps from 2 to 25

Table 25: Ablation of the flow-matching generative process on Vorticity. We vary the ODE solver and the number of integration steps used at sampling time. Relative L^2 error (lower is better).

Ablation	Configuration	Rel. L^2
Solver	Euler	1.62e-1
	Midpoint	1.67e-1
	RK4	1.68e-1
	DOPRI5	1.71e-1
# Steps	2 steps	1.72e-1
	5 steps	1.69e-1
	10 steps	1.68e-1
	25 steps	1.67e-1

brings only marginal gains. These results indicate that our flow-matching generative process is stable with respect to the choice of ODE solver and discretization.

G Visualizations

We provide qualitative visualizations of ECHO's generated trajectories and a comparison with ground truth trajectories.

G.1 Vorticity

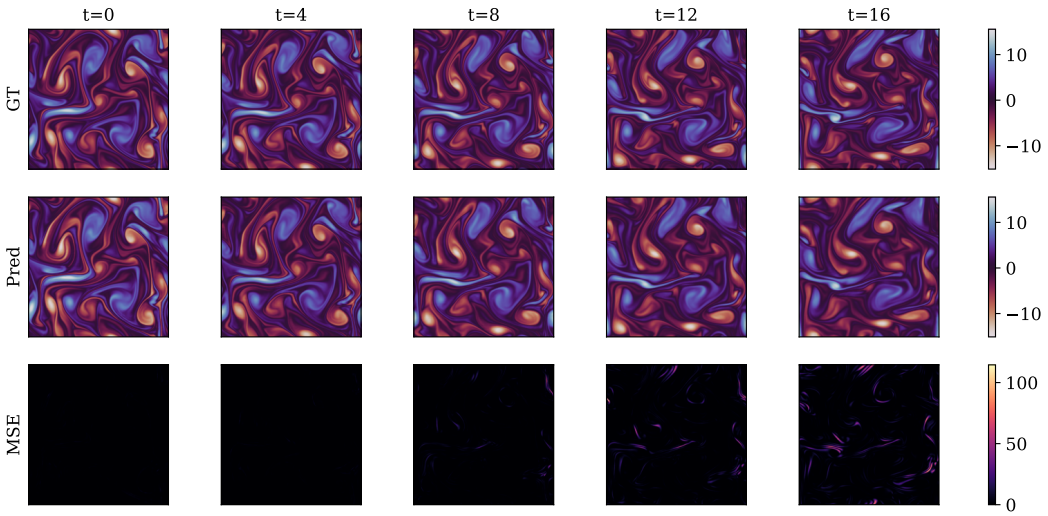


Figure 20: Generation using ECHO's framework on the Vorticity dataset. We generate 1024×1024 spatial points per frame. (Top) ground truth trajectory, (Middle) ECHO's prediction, (Bottom) Error.

G.2 Gray-Scott

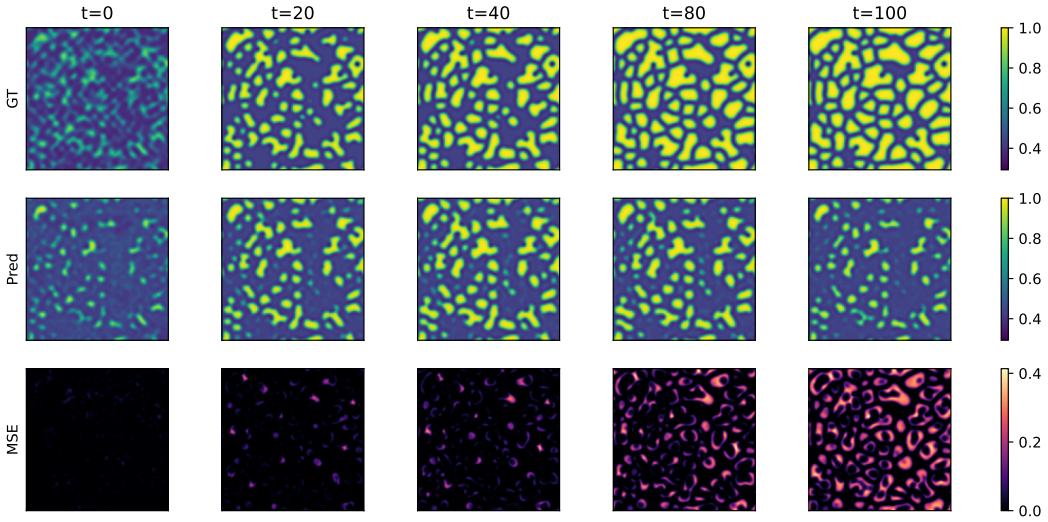


Figure 21: Generation using ECHO's framework on the Gray-Scott dataset. (Top) ground truth trajectory, (Middle) ECHO's prediction, (Bottom) Error.

G.3 Rayleigh-Benard

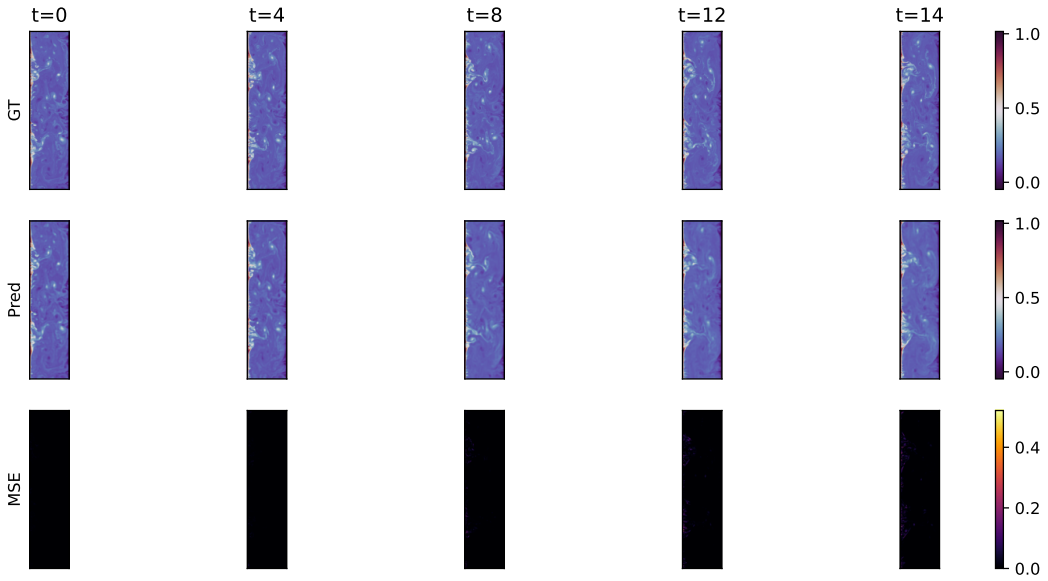


Figure 22: Generation using ECHO's framework on the Rayleigh-Benard dataset. (Top) ground truth trajectory, (Middle) ECHO's prediction, (Bottom) Error.

G.4 Active-Matter

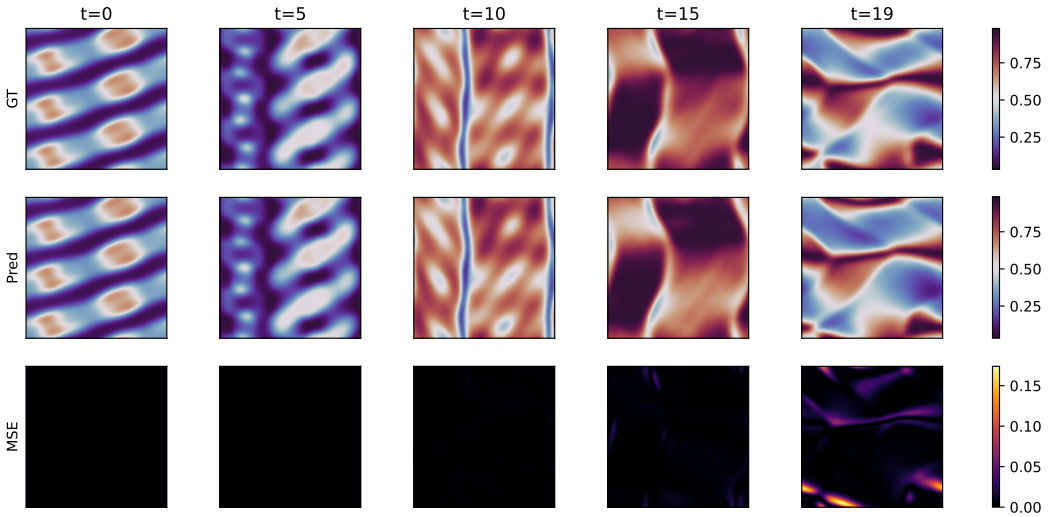


Figure 23: Generation using ECHO's framework on the Active Matter dataset. (Top) ground truth trajectory, (Middle) ECHO's prediction, (Bottom) Error.

G.5 Acoustic Scattering Maze

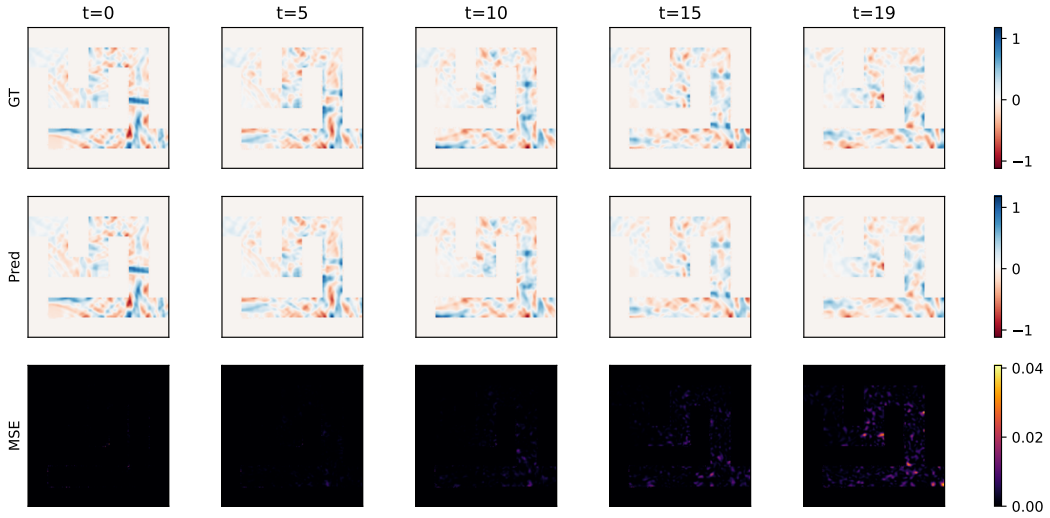


Figure 24: Generation using ECHO's framework on the Acoustic Scattering Maze dataset. (Top) ground truth trajectory, (Middle) ECHO's prediction, (Bottom) Error.

G.6 Eagle

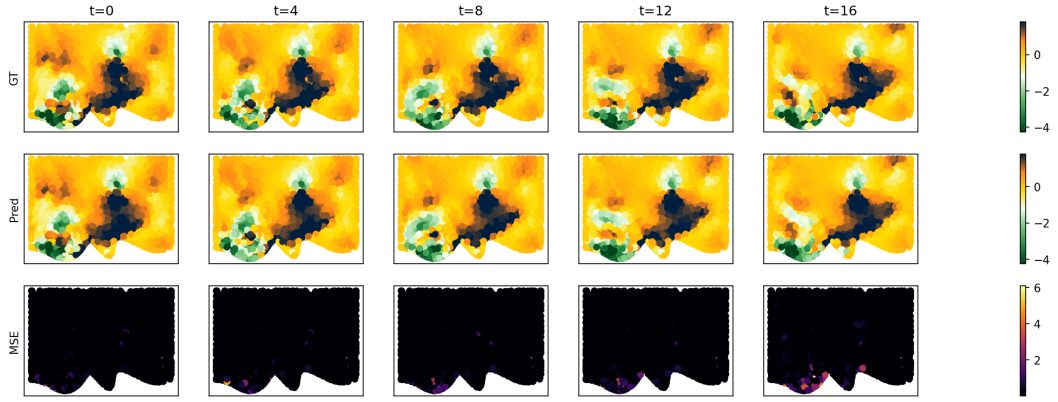


Figure 25: Generation using ECHO's framework on the Eagle dataset. (Top) ground truth trajectory, (Middle) ECHO's prediction, (Bottom) Error.

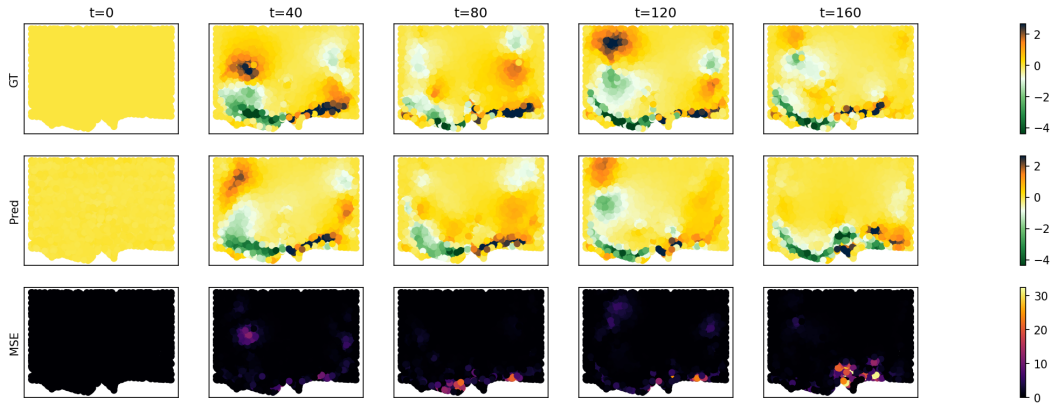


Figure 26: Generation using ECHO's framework on the Eagle dataset - long trajectory generation. (Top) ground truth trajectory, (Middle) ECHO's prediction, (Bottom) Error.

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: The claims in the abstract are detailed in the paper and specifically empirically demonstrated in the extensive experiments provided in section 4 and section F.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: A discussion on the limitations is provided in section 5.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: The paper does not include any theoretical proof.

Guidelines:

- The answer [N/A] 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: We provide a detailed review of our architecture and all hyper-parameters in sections C to E to ensure reproducibility of the results. Additionally, our experimental settings are clearly detailed for each experiment. Code will be released upon acceptance. Finally, we provide a reproducibility statement to specify the GPUs used.

   Guidelines:

   - The answer [N/A] 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: All the baselines and dataset are properly referenced and publicly available. We will release the code of our method upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Our experiments are introduced in section 4. All the hyper parameters used are provided in section E.

Guidelines:

- The answer [N/A] 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: [No]

Justification: We did not provide error bars because it would have been too computationally expensive, as each main experiment requires to train several models.

Guidelines:

- The answer [N/A] 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: We specify in the Reproducibility section the GPU type we used as well as an estimation of the global computation time required for the development and experiments. Moreover, in section E, we provide the computational budget considered for each baseline.

   Guidelines:

   - The answer [N/A] 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: Authors have reviewed the NeurIPS code of ethics and followed it.

   Guidelines:

   - The answer [N/A] 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: Our paper is foundational research, but we discussed the societal impact in the Impact Statement section, as our model can be applied to a wide range of scenarios.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not provide dataset or pre-trained checkpoints of models for this paper.

    Guidelines:

    - The answer [N/A] 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 carefully cited the original source of every dataset and code we used.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: Code will be released upon acceptance and we provide all the details in sections C and E.

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used for polishing writing and were not used for the core contribution.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.