Hierarchical Implicit Neural Emulators

Ruoxi Jiang[1,5] **Xiao Zhang**[1] **Karan Jakhar**[3,6] **Peter Y. Lu**[4,7]
Pedram Hassanzadeh[3] **Michael Maire**[1] **Rebecca Willett**[1,2]
[1]Department of Computer Science, The University of Chicago
[2]Department of Statistics, The University of Chicago
[3]Department of Geophysical Sciences, The University of Chicago
[4]Department of Physics, The University of Chicago
[5]Artificial Intelligence Innovation and Incubation Institute, Fudan University
[6]Department of Mechanical Engineering, Rice University
[7]Department of Electrical and Computer Engineering, Tufts University

Abstract

Neural PDE solvers offer a powerful tool for modeling complex dynamical systems, but often struggle with error accumulation over long time horizons and maintaining stability and physical consistency. We introduce a multiscale implicit neural emulator that enhances long-term prediction accuracy by conditioning on a hierarchy of lower-dimensional future state representations. Inspired by the stability properties of numerical implicit time-stepping methods, we developed an approach that leverages predictions several steps ahead in time at increasing compression rates for next-timestep refinements. By actively adjusting the temporal downsampling ratios, our design enables the model to capture dynamics across multiple granularities and enforce long-range temporal coherence. Experiments on turbulent fluid dynamics show that our method achieves high short-term accuracy and produces long-term stable forecasts, significantly outperforming non-hierarchical autoregressive baselines while adding minimal computational overhead. The codebase is available at this link[1].

1 Introduction

Machine learning (ML)-based surrogate modeling of dynamical systems has spurred great interest in recent years due to transformative applications in climate modeling [1–5], molecular dynamics [6–8], cosmology [9–11], and beyond. These surrogate models or emulators are often orders of magnitude faster than classical numerical solvers and can be trained to directly emulate real-world data. Although existing developments in neural surrogate models have made notable strides in short-term prediction, often benchmarked using one-step mean square error (MSE), their ability to deliver stable long-term predictions remains a critical challenge [12–14]. For scientific practitioners, achieving a long-term stable forecast is valuable and expands the potential for research use, especially where classical numerical solvers are too costly to run. For example, climate scientists can use accurate long-term emulators to predict the probability of extreme events [15]. However, the error accumulation in unrolling the predictions for complex, nonlinear, and multiscale dynamical systems often causes existing surrogate models to degrade catastrophically during long-term forecasting, resulting in unphysical collapsed or exploding solutions [14, 16].

Recent efforts to address long-term stability explore two primary avenues. From the temporal evolution perspective, frameworks centered on multi-step ahead prediction aim to shift focus from single-step accuracy to extended horizons by training models on multi-step targets [17–20]. Yet,

*Correspondence to: roxie_jiang@fudan.edu.cn, zhang7@uchicago.edu, and willett@uchicago.edu.
[1]https://github.com/roxie62/Hierarchical-Implicit-Neural-Emulators

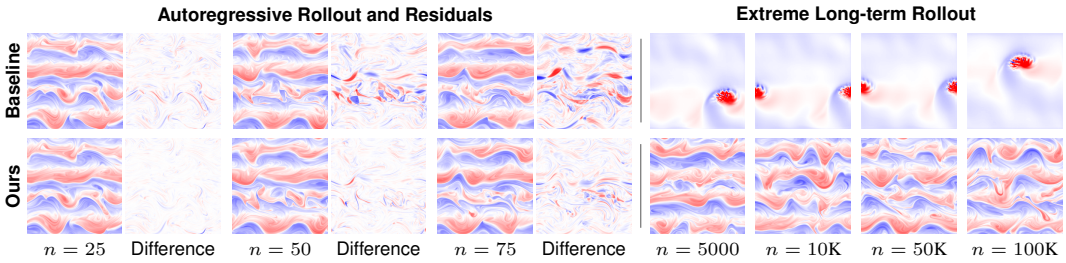


Figure 1: In a chaotic, turbulent system, our emulator achieves accurate short-to-mid-term rollout predictions (*left*). Even over extremely long sequences (up to 10^5 emulation steps, *right*), it captures the physical jet patterns, while baseline autoregressive methods quickly drift and break down.

such methods can underperform even one-step ahead autoregressive baselines for short-term complex dynamics [21]. From the spatial domain perspective, architectures that use frequency decompositions or long-term statistics as training targets advance the performance of surrogate modeling to better handle the multiscale structure of the data [22–26]. However, they often face a trade-off between short-term accuracy and long-term robustness, or require additional physical knowledge.

In this work, we draw inspiration from numerical analysis and reframe neural surrogate modeling through the lens of time-stepping methods for solving differential equations. Viewing the conventional autoregressive models from this perspective, we can interpret their failure patterns as analogous to the instability of explicit schemes (e.g., forward Euler), especially for larger timesteps and in stiff systems. To improve stability (Fig. 1), we propose to enhance the autoregressive system based on an analogy with the implicit time-stepping methods, which are known for their unconditional stability in challenging integration scenarios [27].

The cornerstone of our approach is the use of *future* frame information to guide next-step predictions, fundamentally departing from existing autoregressive frameworks constrained to the use of past temporal states. While inspired by numerical implicit schemes that use iterative solvers to refine predictions, our framework avoids the corresponding computational overhead by effectively performing the iterative refinement steps in parallel. This is achieved by simultaneously forecasting a hierarchy of future states during autoregressive rollout (Fig. 2), resulting in nearly the same number of forward passes compared to the one-step ahead baseline model. Furthermore, unlike prior multi-step ahead approaches [17–19], our framework actively compresses future states into a hierarchical representation, balancing accuracy and computational tractability while enabling feedback from future time frames. On 2D turbulence with multiscale and chaotic structure, our method demonstrates greatly enhanced long-term stability compared to the baseline models (Fig. 1, *right*), while introducing a minimal amount of additional computational overhead.

We summarize our contributions as follows:

- **Neural emulation with an implicit schema and a hierarchical structure.** We propose a novel emulator that uses an augmented state built from a hierarchy of coarse-grained representations of future states. This approach captures multiscale interactions and — by autoregressively applying the emulator to the augmented state — efficiently mimics the iterative refinement mechanism of implicit solvers, achieving better accuracy and stability.
- **Accurate simulation of chaotic systems.** Our approach is well-suited for turbulent flows, which are chaotic and have complex multiscale structures. On a 2D Navier-Stokes system with Reynolds number of 10^4, our model achieves substantial improvements — reducing mean squared error by over 50% for predictions 25 to 50 steps, compared to the standard 1-step ahead baseline.
- **Stability for extremely long rollouts that are $10\times$ the training length.** Our emulator exhibits strong stability over extremely long sequences. For the 2D Navier-Stokes system at 10^4 Reynolds number, it rolls out up to $\mathbf{2 \times 10^4}$ steps (one time the length of the training set) without empirical failure. Even at $\mathbf{2 \times 10^5}$ steps (ten times the length of the training set), it maintains a low failure rate of just 7.0%, far outperforming existing baselines, which typically fail much earlier and for all initial conditions.

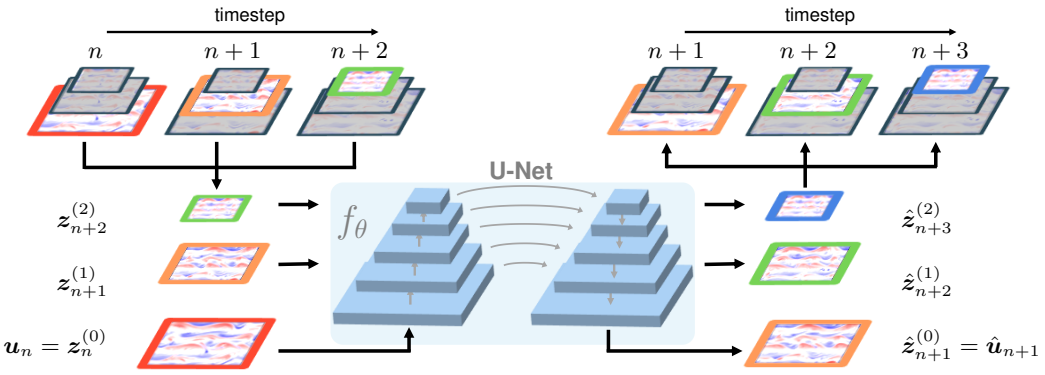


Figure 2: **Diagram of our hierarchical implicit neural emulator.** Our model approximates the dynamics $\boldsymbol{u}$ using an implicit formulation with hierarchical latent representations. For each trajectory $\boldsymbol{u}_n$, we augment its state with multiscale latent features $\boldsymbol{z}_n^{(l)}$, obtained by downsampling $\boldsymbol{u}_n$. The model predicts the future state $\hat{\boldsymbol{u}}_{n+1}$ and long-horizon latent representations $\hat{\boldsymbol{z}}^{(1)}_{n+2}$, $\hat{\boldsymbol{z}}^{(2)}_{n+3}$, enabling it to capture long-term dynamics. Following the implicit schema, the model conditions on both the past trajectory $\boldsymbol{u}_n$ and future latent variables $\boldsymbol{z}^{(1)}_{n+1}, \boldsymbol{z}^{(2)}_{n+2}$ during training, while using predictions of future states computed in the previous step of an autoregressive rollout during inference, providing richer context to effectively mitigate error accumulation for long-term predictions.

2 Related Work

Hierarchical networks. Many physical systems exhibit patterns across spatial and temporal scales. To effectively emulate such dynamical systems, the model architecture should reflect their inherent hierarchical structure. A U-Net [28], for instance, introduces skip connections between different spatial resolutions, enabling effective reasoning across both high-level semantics and fine-grained details. Many subsequent approaches similarly integrate hierarchical structure into a system's data representations or computational elements. Feature Pyramid Networks [29] fuse multiscale features to improve prediction accuracy. Multiscale Vision Transformers [30] construct hierarchical representations tailored for attention mechanisms. Inspired by multigrid solvers [31], multigrid CNNs [32, 33] maintain multiscale representations at each layer and introduce cross-scale convolution operations to enable efficient information routing via bidirectional connections between adjacent scales. Hierarchical VAEs (HVAEs) [34] extend latent variables across multiple scales, significantly improving generative model quality. However, HVAEs often face challenges such as high training variance and instability. Nested Diffusion Models [35] address similar challenges by pretraining and freezing the hierarchical latents, then training a collection of conditional diffusion models to generate each level of the latent hierarchy conditioned on those above, concluding with image generation.

Neural PDE solvers. Neural PDE solvers offer a data-driven alternative to traditional numerical methods, with promising applications in real-world applications, such as weather forecasting (e.g., FourCastNet [1], GenCast [36]). These methods fall into two main classes: (1) neural operators, which map between function spaces (e.g., FNO [22], DeepONet [37]), and (2) grid-based architectures, which operate on a discretized grid or mesh (e.g., U-Nets [28]). While neural operators promise mesh-agnostic inference, grid-based methods remain prevalent due to their simplicity and popularity in computer vision. Although our current implementation assumes grid-structured inputs, the framework can be naturally extended to mesh-based representations, as demonstrated in prior work [38–40].

A major challenge for both is long-term stability, where small errors amplify over time. Stabilization efforts include: (1) data-driven methods like pushforward training [18] and PDE-Refiner [41], which improve rollouts but suffer from instability and high computational cost; (2) physics-informed strategies such as multi-time stepping [42], which embed physical priors but limit data-driven flexibility; and (3) architectural advances like FNOs [22] and Wavelet Neural Operators [23], which improve spatial modeling but struggle with temporal coherence and optimization. Despite progress, current approaches fall short in handling multiscale chaotic systems, such as turbulent flows, where spatiotemporal complexity demands both efficient learning and robust long-horizon predictions.

3 Problem Formulation

Dynamical systems. Consider a nonlinear dynamical system evolving over time as:

$$\frac{\partial u}{\partial t} = f(u, x, t, \nabla_x u, \nabla_x^2 u \dots), \quad u(x, 0) = u_0, \tag{1}$$

where f encodes the unknown governing physics. Let $u(x, t) \in \mathbb{R}$ denote the continuous trajectory of dynamics in function space $\mathcal{U}$, and $\boldsymbol{u}_t \in \mathbb{R}^m$ represent discretized snapshots (i.e., sample from a finite spatial grid with m points) at time t. To numerically approximate the dynamics, we adopt discrete methods (e.g., finite differences, spectral methods), transforming the PDE into a system of ODEs:

$$\frac{d\boldsymbol{u}_t}{dt} = \tilde{f}(\boldsymbol{u}_t, t), \quad \boldsymbol{u}_0 \in \mathbb{R}^m, \tag{2}$$

where $\tilde{f}$ corresponds to the temporal dynamics at discretized states. Given $N + 1$ consecutive observations $\{\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_N\}$, *our goal* is to learn a neural emulator f_θ that approximates the underlying dynamics and predicts future states. The emulator is parameterized by θ.

Autoregressive models. Autoregressive methods learn a transition operator $f_\theta : \mathbb{R}^m \to \mathbb{R}^m$ to iteratively predict next state $\hat{\boldsymbol{u}}_{n+1}$ from $\boldsymbol{u}_n$:

$$\hat{\boldsymbol{u}}_{n+1} = f_\theta(\boldsymbol{u}_n), \tag{3}$$

and chain predictions via recursive rollouts:

$$\hat{\boldsymbol{u}}_{n+k} = f_\theta(f_\theta(f_\theta \dots (\boldsymbol{u}_n))) \quad \text{for } k \text{ steps.} \tag{4}$$

While efficient and accurate for short-term predictions, autoregressive methods often suffer from error accumulation over long horizons, where small discrepancies compound over time [18, 14]. Our approach adheres to this autoregressive framework, but introduces mechanisms to mitigate these long-term errors, as described in Section 4.

4 Method

In this section, we show how to improve short-term forecast and preserve long-term coherence in an autoregressive model by building an implicit multiscale framework.

4.1 Implicit Neural Emulator

Explicit *v.s.* implicit methods. For the semi-discretized system in Eqn. 2, explicit time-stepping methods [43] compute future states directly through:

$$\boldsymbol{u}_{n+1} \approx F(\boldsymbol{u}_n), \tag{5}$$

where the mapping $F(\cdot)$ represents an explicit update rule depending only on the current state $\boldsymbol{u}_n$. For instance, in the *forward* Euler method [44], it takes the form:

$$F(\boldsymbol{u}_n) := \boldsymbol{u}_n + \Delta t \tilde{f}(\boldsymbol{u}_n) \tag{6}$$

to compute an estimate of $\boldsymbol{u}_{n+1}$. This approach can be connected to the autoregressive learning seen in Eqn. 3, where predictions rely solely on past states. And similar to autoregressive methods, it is known to suffer from the instability using a larger time step Δt or facing systems with stiffness [45], where adaptive and higher-order variants are usually required to ensure stability of the solver, which substantially increases the computational cost.

Implicit methods are designed to solve these issues: they leverage both current states $\boldsymbol{u}_n$ and future states $\boldsymbol{u}_{n+1}$ to solve the following implicit equation:

$$\boldsymbol{u}_{n+1} = F(\boldsymbol{u}_n, \boldsymbol{u}_{n+1}). \tag{7}$$

Here $F(\boldsymbol{u}_n, \boldsymbol{u}_{n+1}) := \boldsymbol{u}_n + \Delta t \tilde{f}(\boldsymbol{u}_{n+1})$ in the *backward* Euler method. Solving $\boldsymbol{u}_{n+1}$ in Eqn. 7 requires extra computations to find the root. A classic example is Newton's method, which iteratively updates the estimate via:

$$\boldsymbol{u}_{n+1}^{i+1} = \boldsymbol{u}_{n+1}^{i} - \frac{F(\boldsymbol{u}_n, \boldsymbol{u}_{n+1}^{i})}{F'(\boldsymbol{u}_n, \boldsymbol{u}_{n+1}^{i})} \tag{8}$$

where i indicates the iteration index. Compared to explicit methods, implicit approaches can accommodate much larger timesteps while maintaining stability, offering computational efficiency for long-time integration.

Implicit neural emulator with two-step prediction. Drawing inspiration from the robustness of implicit solvers, we propose a neural network architecture that combines the stability of implicit methods with the simplicity of explicit methods. Our goal is to address the compounding error typically seen in autoregressive models by introducing a structure that implicitly reasons about future states. Rather than solving the implicit equation in Eqn. 7 directly — which would entail iterative root-solving at each step — we adopt a relaxed version. We introduce a latent variable $\boldsymbol{z}_{n+1} = T(\boldsymbol{u}_{n+1})$, which represents an abstract encoding of the future state $\boldsymbol{u}_{n+1}$. This latent variable can be directly computed during training, but must be predicted at test time, leading to the following formulation:

$$\hat{\boldsymbol{u}}_{n+1}, \hat{\boldsymbol{z}}_{n+2} = f_\theta\left(\boldsymbol{u}_n, \boldsymbol{z}_{n+1}\right), \tag{9}$$

where the network is trained to simultaneously predict the next physical state $\hat{\boldsymbol{u}}_{n+1}$ and the latent representation $\hat{\boldsymbol{z}}_{n+2}$ that encodes information about a future state.

This architecture mirrors the iterative refinement process used in implicit solvers. Our model is given an initial approximation of state $n+1$ represented by $\boldsymbol{z}_{n+1}$ and then refines this approximation by applying $f_\theta(\boldsymbol{u}_n, \boldsymbol{z}_{n+1})$ to produce the predicted physical state $\hat{\boldsymbol{u}}_{n+1}$. By also providing $\hat{\boldsymbol{z}}_{n+2}$ as an auxiliary output, the model is able to perform this refinement process as part of the standard autoregressive rollout across timesteps, effectively treating $(\boldsymbol{u}_n, \boldsymbol{z}_{n+1})$ as an augmented state variable.

In practice, we choose the transformation T to be a downsampling or coarse-graining operation. Predicting a coarse-grained $\hat{\boldsymbol{z}}_{n+2}$ rather than the full physical state $\hat{\boldsymbol{u}}_{n+2}$ encourages the emulator to first capture large-scale spatial features and, as shown in our experiments, leads to better prediction accuracy. We will elaborate on the design of the transformation T in a later subsection.

4.2 Hierarchical Multi-step Prediction

The architecture above naturally extends to a hierarchical multi-step modeling framework in which predictions are conditioned on multiple latent representations of anticipated future states. We denote these representations as $\boldsymbol{z}_m^{(l)} = T^{(l)}(\boldsymbol{u}_m)$, where l indexes increasing levels of abstraction and we assume $\boldsymbol{u}_m = \boldsymbol{z}_m^{(0)}$. The model is then trained to predict across L hierarchical levels as follows:

$$\hat{\boldsymbol{u}}_{n+1}, \hat{\boldsymbol{z}}_{n+2}^{(1)}, \ldots, \hat{\boldsymbol{z}}_{n+L}^{(L-1)} = f_\theta\left(\boldsymbol{u}_n, \boldsymbol{z}_{n+1}^{(1)}, \ldots, \boldsymbol{z}_{n+L-1}^{(L-1)}\right). \tag{10}$$

As shown in Figure 2, our model effectively operates in an augmented state space formed by the concatenation of the current state with future latent states of increasing levels of abstraction. In practice, this simple strategy proves scalable and robust, and easily extensible to deeper hierarchies.

This generalized framework enhances the analogy to the multi-step iterative refinement used in implicit solvers while also enabling the model to capture broader temporal dynamics, which helps mitigate error accumulation by informing each prediction with a richer view of future latent dynamics.

Leveraging future context in a hierarchical fashion. Using future context offers two complementary benefits. From an *encoding* standpoint, it allows the model to process input as a structured sequence where immediate states like $\boldsymbol{u}_n$ retain fine-grained detail, while distant latents such as $\boldsymbol{z}_{n+l}^{(l)}$ provide coarse-scale insights like aggregate dynamics or long-term trends. This mirrors the philosophy of multi-step implicit methods like Adams–Moulton [46], which integrate information from multiple past states to produce more accurate future estimates, though with a slight distinction that we are conditioning on the future frames, while being able to extend to include the history states as well.

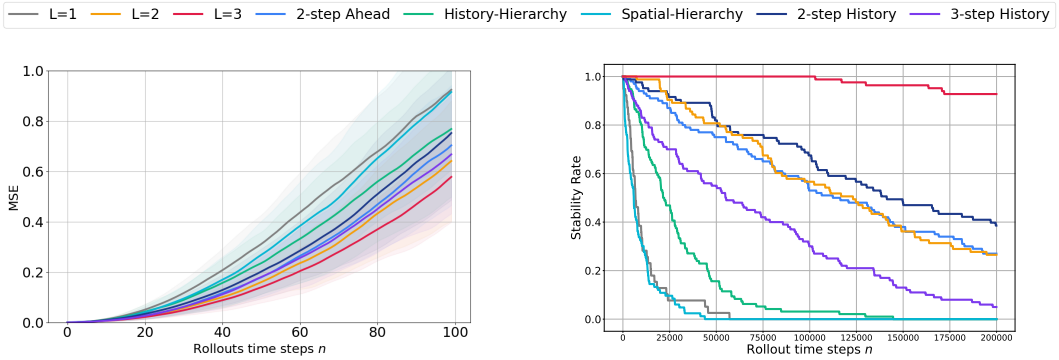
From a *decoding* standpoint, learning to predict distant states becomes progressively harder due to increased uncertainty and error amplification. By supervising the model across multiple levels of abstraction, we encourage a balanced learning process where both local precision and global structure are prioritized. Viewed through the lens of implicit solvers, this can be interpreted as executing L steps of iterative refinement, distributed across temporal frames and abstraction levels.

4.3 Training Details

Choosing abstract transformations. A key component of training our model is the choice of the transformations $T^{(l)}$, which map physical states $\boldsymbol{u}_n$ to latent representations $\boldsymbol{z}_n$. There are

	Confi. Name	Model	Config. Name	Model
Ours	$L = 2$	$\hat{\boldsymbol{u}}_{n+1}, \hat{\boldsymbol{z}}^{(1)}_{n+2} = f_\theta(\boldsymbol{u}_n, \boldsymbol{z}^{(1)}_{n+1})$	$L = 3$	$\hat{\boldsymbol{u}}_{n+1}, \hat{\boldsymbol{z}}^{(1)}_{n+2}, \hat{\boldsymbol{z}}^{(2)}_{n+3} = f_\theta(\boldsymbol{u}_n, \boldsymbol{z}^{(1)}_{n+1}, \boldsymbol{z}^{(2)}_{n+2})$
Baseline	$L = 1$	$\hat{\boldsymbol{u}}_{n+1} = f_\theta(\boldsymbol{u}_n)$	Spatial Hierarchy	$\hat{\boldsymbol{u}}_{n+1}, \hat{\boldsymbol{z}}^{(1)}_{n+1}, \hat{\boldsymbol{z}}^{(2)}_{n+1} = f_\theta(\boldsymbol{u}_n, \boldsymbol{z}^{(1)}_{n}, \boldsymbol{z}^{(2)}_{n})$
	2-Step Ahead	$\hat{\boldsymbol{u}}_{n+1}, \hat{\boldsymbol{u}}_{n+2} = f_\theta(\boldsymbol{u}_n)$	History Hierarchy	$\hat{\boldsymbol{u}}_{n+1} = f_\theta(\boldsymbol{u}_n, \boldsymbol{z}^{(1)}_{n-1}, \boldsymbol{z}^{(2)}_{n-2})$
	2-Step History [19]	$\hat{\boldsymbol{u}}_{n+1} = f_\theta(\boldsymbol{u}_{n-1}, \boldsymbol{u}_n)$	3-Step History [19]	$\hat{\boldsymbol{u}}_{n+1} = f_\theta(\boldsymbol{u}_{n-2}, \boldsymbol{u}_{n-1}, \boldsymbol{u}_n)$

Table 1: **Model and baseline configurations.** We conduct experiments with our model designs ($L = 2, L = 3$) and several baseline configurations for comprehensive analysis.



(a) Performance of short-term estimation measured by mean squared error (MSE)

(b) Stability rate for long-term rollout, up to 2×10^5 steps

Figure 3: **Short-term accuracy vs. long-term robustness.** *Left:* MSE trend over a 100-step autoregressive rollout. Solid lines indicate the average performance, and shaded regions represent one standard deviation. *Right:* We evaluate the stability rate over 10 times the training sequence length across 100 trials with various initial conditions. Stability is defined by maintaining physical statistics (conserved kinetic energy). Our method with $L = 3$ achieves a 93% stability rate, significantly outperforming all baselines, with the next best, 2-Step History, at just 38%, indicating superior short-term accuracy and long-term robustness.

several viable options for $T^{(l)}$, such as a learnable encoder or a predefined information-reduction function (e.g., spatial pooling). While a learned encoder offers expressiveness, it can introduce instability during training due to the constantly shifting latent representations. Instead, we focus on the multiscale structure of spatiotemporal systems and choose $T^{(l)}$ to be a set of fixed spatial downsampling transformations:

$$\boldsymbol{z}^{(l)}_n = T^{(l)}(\boldsymbol{u}_n) := \text{DownSample}(\boldsymbol{u}_n, r^l), \tag{11}$$

Here r^l denotes the predefined downsampling rate for the l-th level with $r^l \leq r^{l+1}$, providing a sequence of increasingly coarse-grained states.

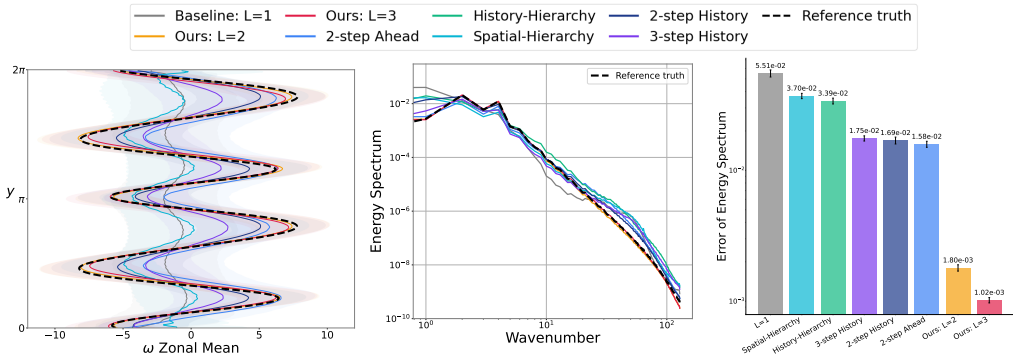
Training objective. Finally, our training loss is designed to supervise both the predicted physical state and the associated abstract latents:

$$\ell(\theta) = d(\hat{\boldsymbol{u}}_{n+1}(\theta), \boldsymbol{u}_{n+1}) + \sum_{l=1}^{L-1} d(\hat{\boldsymbol{z}}^{(l)}_{n+l}(\theta), \boldsymbol{z}^{(l)}_{n+l}), \tag{12}$$

where $d(\cdot)$ denotes a distance metric such as $l1$ or $l2$ loss for simplicity, though this could be replaced with domain-specific alternatives. This objective ensures that the model's predictions remain aligned with both immediate and long-term dynamics, across all abstraction levels.

5 Experiments

In this section, we benchmark our implicit modeling framework for 2D turbulence prediction. This canonical flow has been extensively used for testing novel ML-based schemes [47–49, 41, 50].



(a) Zonal Mean (b) Energy Spectrum (c) Error of Energy Spectrum

Figure 4: **Further investigation of long-term stability of 2×10^5 steps (10$\times$ the training length) rollout.** *(a):* Time-averaged zonal mean of vorticity comparing ground truth (dashed black lines) with emulator runs. Only our systems with $L = 2$ and $L = 3$ accurately follow the ground truth trend, with the reference truth lies well within error bars, demonstrating that our emulator accurately captures the system's mean statistics. In contrast, other methods significantly deviate, indicating a substantial drift and failure to preserve structure. *(b,c):* Spectrum of long-term rollout.

Navier-Stokes. We focus on the dimensionless vorticity-streamfunction ($\omega - \psi$) formulation of the incompressible Navier-Stokes equations in a 2D $x - y$ domain [47, 48]:

$$\frac{\partial \omega}{\partial t} + \mathcal{N}(\omega, \psi) = \frac{1}{Re}\nabla^2\omega - \chi\omega + f + \beta v_y, \tag{13}$$

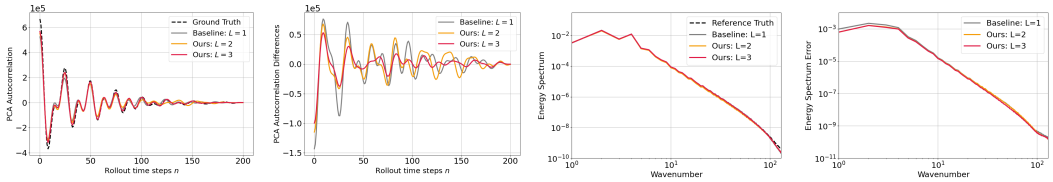
where $\nabla^2\psi = -\omega, \boldsymbol{v} = (v_x, v_y)$ is velocity, and $\omega = \nabla \times \boldsymbol{v}$. $\mathcal{N}(\omega, \psi)$ is the Jacobian and represents non-linear advection. The flow is defined by a Reynolds number $Re = 10^4$, time-constant forcing f at a given wavenumber, and a Rayleigh drag $\chi = 0.1$. The Coriolis parameter, $\beta = 20$, induces zonal jets characteristic of geophysical turbulence, mimicking the influence of Earth's rotation on atmospheric and oceanic flows [51]. The domain is doubly periodic with length $L = 2\pi$. Training data is generated using "py2d" [52] on a 512×512 grid, with 18,000 ω snapshots. For analysis in this paper, the data is downsampled to a 256×256 grid, except in Section 5.3, which discusses ablation studies on different datasets and resolutions.

Architecture. We adopt the UNet design following Dhariwal and Nichol [53] as our base model, following the standard setup with using a structured dataset. Specifically, our model has an encoder composed of 5 downsampling blocks that reduce the input to an $8 \times 8 \times 256$ feature map at the bottleneck. The decoder mirrors the encoder structure to reconstruct the output. To improve the model's ability to capture high-frequency details, we incorporate Fourier layers [22] into each block to better handle high-frequency information (see details in Appendix A.3).

Given that our design must accommodate images at multiple resolutions, we align with the UNet's hierarchical information flow: lower-resolution inputs are introduced at the earlier encoder stages, while decoder blocks operating at matching resolutions are used to produce the final output. This structure ensures that the most abstract features are captured at the encoder's bottleneck, promoting a coherent and structured flow of information throughout the network.

Model configurations and baseline setup. Our experiments primarily focus on designs with two-level and three-level hierarchies, indicated by $L = 2$ and $L = 3$. For the input with 256×256 resolution, we choose $r^1 = 8$ and $r^2 = 32$ as our default downsampling rate for latent variables $\boldsymbol{z}^{(1)}_{n+1}, \boldsymbol{z}^{(2)}_{n+2}$ respectively. To rigorously assess our model's performance, we also propose several baseline methods that diverge from our formulation, including different setups of input/output hierarchy configurations, as detailed in Table 1.

Hierarchical autoregressive rollout. To enable autoregressive rollout given only the initial condition $\boldsymbol{u}_n$, we introduce specially-designed corner-case inputs, corresponding to $L - 1$ higher-level latent slots, during both training and inference. During the prediction, with a small probability p (see details in Appendix A.3), we sample training instances in which the model receives a partially missing hierarchy of latent inputs and is tasked to reconstruct the missing parts. Concretely, for our $L = 2$ levels design, we present the model with $[\boldsymbol{u}_n, \boldsymbol{0}]$, where $\boldsymbol{0}$ denotes the zero-filled tensor matching the



(a) PCA autocorrelation along rollout steps (b) PCA autocorrelation errors along rollout steps (c) Averaged energy spectrum over wavenumber (d) Energy spectrum error over wavenumber

Figure 5: **Ablation studies on the design of the hierarchy.** We evaluate our system across different hierarchy levels L and visualize the results using: *(a)* PCA autocorrelation and *(c)* energy spectrum averaging over 200 rollout steps, with corresponding deviations from the ground truth shown in *(b)* and *(d)*, respectively. Our hierarchical design with $L = 3$ outperforms the baseline $L = 1$, achieving better alignment with temporal structure and high-frequency components.

	1	25	50	75	100
$r^1 = 1$	9.900e-04 (2.143e-04)	5.865e-02 (2.890e-02)	2.103e-01 (8.888e-02)	4.470e-01 (1.762e-01)	7.738e-01 (2.520e-01)
$r^1 = 2$	1.747e-03 (3.942e-04)	1.146e-01 (6.328e-02)	3.774e-01 (1.797e-01)	7.082e-01 (3.088e-01)	1.023e+00 (3.352e-01)
$r^1 = 4$	7.735e-04 (1.954e-04)	5.651e-02 (3.199e-02)	1.998e-01 (1.074e-01)	4.225e-01 (1.979e-01)	7.280e-01 (2.522e-01)
$r^1 = 8$	5.248e-04 (1.148e-04)	**4.024e-02 (1.920e-02)**	**1.606e-01 (6.496e-02)**	**3.731e-01 (1.509e-01)**	**6.547e-01 (2.440e-01)**
$r^1 = 16$	**5.155e-04 (1.138e-04)**	4.898e-02 (2.011e-02)	2.039e-01 (8.009e-02)	4.416e-01 (1.789e-01)	7.399e-01 (2.658e-01)

Table 2: **Ablation study on downsampling ratio (r^1) for our $L = 2$ model on 256×256 resolution data with jet.** We evaluate model performance by computing roll-out mean squared error (MSE) for downsampling ratios $r^1 = 1, 2, 4, 8, 16$, and present the mean (standard deviation) over 100 trials with varied initial conditions. Notably, $r^1 = 8$ exhibits the minimal error, indicating an ideal balance between preserving contextual information and mitigating estimation challenges.

shape of the higher hierarchy latent $\hat{\boldsymbol{z}}^{(1)}$, and train it to predict $\hat{\boldsymbol{z}}^{(1)}_{n+1}$. This ensures that the model learns to infer higher-level latent states from just the initial condition. At the evaluation time, each method is provided only with the initial condition $\boldsymbol{u}_n$. In the hierarchy $L = 2$, we first input $[\boldsymbol{u}_n, \boldsymbol{0}]$ to obtain $\hat{\boldsymbol{z}}^{(1)}_{n+1}$. Then, using the full spatial-temporal hierarchy states $[\boldsymbol{u}_n, \hat{\boldsymbol{z}}^{(1)}_{n+1}]$, we continue with autoregressive rollout.

5.1 Short-term Accuracy and Long-term Stability

We assess our model's performance under two scenarios: (1) Short-term predictions, up to 100 steps using mean squared error (MSE); and (2) Extremely long-term predictions, generating up to 2×10^5 steps to evaluate the alignment of predicted and true physical statistics.

Evaluation on short-term estimation. Figure 3(a) presents MSE results comparing our method with various baselines. All methods show similar 1-step MSE, yet the 2-Step Ahead Baseline performs poorly (see Table 6 in Appendix A.1). Our models with $L = 3$ and $L = 2$ consistently provide accurate estimates, surpassing all baselines despite challenges in recursive prediction error accumulation. Baselines using multiple input or output frames (*e.g.*, 3-Step History or 2-Step Ahead) perform better for *mid-term* rollout, highlighting the importance of processing temporal sequences in autoregressive models. In comparison, basic hierarchical baselines (*e.g.*, History-Hierarchy or Spatial-Hierarchy) provide minimal gains, suggesting that temporal hierarchy is crucial.

Evaluation on long-term estimation. Over very long rollouts, mean squared error (MSE) becomes an unreliable metric for evaluating autoregressive systems. Instead, we verify if the predictions align with the true physical statistics, such as the system's energy given by $E = \frac{1}{2}\left(v_x^2 + v_y^2\right)$. For a statistically stationary turbulent system, like the one studied here, this energy stays around a constant mean due to the balance between energy input by the deterministic forcing and its dissipation via viscosity and the linear drag in equation 13 [54].

Stability rate. We use a simple binary energy-based approach to check if a prediction adheres to correct physical patterns. By calculating the mean and standard deviation of energy from a 200K-step true trajectory, we ensure all true trajectory energy values fall within 4 standard deviations of the mean. A prediction is deemed stable if its energy remains within 5 standard deviations of the true mean (see

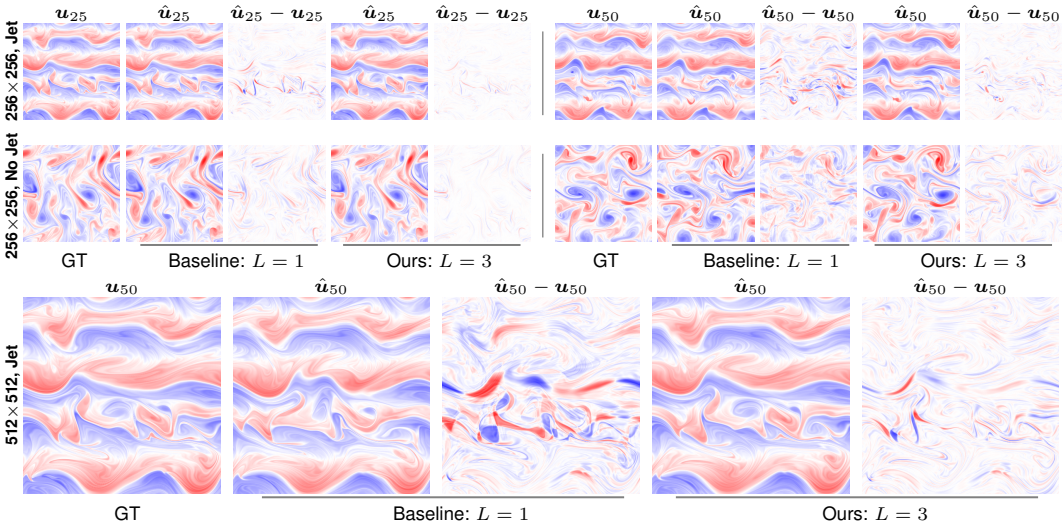


Figure 6: **Visualization of rollout estimation across multiple datasets.** We apply our approach to three different flows: (1) $Re = 10^4$, 256×256 resolution featuring zonal jets, (2) $Re = 5 \times 10^3$, 256×256 resolution without zonal jets, and (3) $Re = 10^4$, 512×512 resolution with zonal jets. Our method ($L = 3$) gives more accurate predictions with lower associated residuals in all three scenarios.

Appendix A.2). Appendix B.2 includes examples with varying energy deviations, illustrating that predictions exceeding this threshold exhibit incorrect physical patterns.

Figure 3(b) presents the stability rates for all methods, based on predictions over 200K steps from 100 initial conditions, evaluated every 100 emulation steps. Our approach with $L = 3$ consistently sustains a high stability rate during rollout, achieving **93.0**% stability through the 2×10^5 steps. This greatly surpasses the closest competitor, the 2-Step History method, which only reaches 38.0%, with 2-Step Ahead and our $L = 2$ following. Most other methods fail at the end of rollouts, indicating their limitations in capturing long-term dynamics. These findings highlight the need for our methods that can handle longer temporal dependencies to ensure robust long-term performance.

Zonal mean and energy spectrum. We extend our evaluation using (1) the zonal mean and (2) the energy spectrum (see definitions in Appendix A.2). The zonal mean reveals the flow's large-scale organization, namely, jet formation and maintenance. In geophysical turbulence, jets are crucial structures resulting from nonlinear advection, planetary rotation, and dissipation. Capturing the zonal mean is vital for verifying that the emulator preserves these structures over time. As seen in Fig. 4(a), our methods ($L = 2$ and $L = 3$) produce zonal mean profiles within error bands, successfully reproducing the jets, unlike baseline ($L = 1$) and other methods. The energy spectrum illustrates kinetic energy distribution over wavenumbers. In Fig. 4(b), $L = 2$ and $L = 3$ exhibit minimal bias, aligning closely with the reference spectrum, even at small wavenumbers. These findings confirm our model's robustness, maintaining long-term stability while accurately reflecting the system's statistical properties.

5.2 Ablation Studies

Ablation on model hierarchy. A key aspect of our model is its ability to process and estimate hierarchical representations of the data. To further examine this design, we analyze PCA autocorrelation to capture temporal structure and the energy spectrum to evaluate frequency-domain accuracy. Computation details are provided in the Appendix A.3. The results are shown in Fig. 5 and Fig. 6. Our model with $L = 3$ preserves temporal patterns more effectively and provides more accurate estimates of high-frequency components compared to $L = 2$ and $L = 1$.

Ablation on downsampling factor. Our system eases the future prediction tasks by initially assessing a downsampled version of the future frame. The downsampling ratio plays an important role in balancing estimation challenges and available information for next frame prediction. We ablate our two-level system across various downsampling rates of $r^1 \in [2, 4, 8, 32]$. Results presented in Table 2 demonstrate that $r^1 = 8$ provides the best performance for nearly all prediction steps.

Rollout Step	1	10	25	50	75	100
$L = 1$ ($\ell_1 + \ell_2$ loss)	5.60e−04 (1.15e−04)	1.29e−02 (4.87e−03)	8.04e−02 (3.45e−02)	3.12e−01 (1.15e−01)	6.19e−01 (2.08e−01)	9.38e−01 (2.88e−01)
$L = 1$ (Sobolev loss)	7.01e−04 (4.07e−04)	1.64e−01 (8.54e−02)	8.76e−01 (2.45e−01)	1.40e+00 (2.30e−01)	1.64e+00 (2.56e−01)	1.69e+00 (2.51e−01)
$L = 3$ ($\ell_1 + \ell_2$ loss)	5.50e−04 (1.20e−04)	6.59e−03 (2.50e−03)	3.37e−02 (1.41e−02)	1.40e−01 (6.16e−02)	3.24e−01 (1.18e−01)	5.92e−01 (1.84e−01)
$L = 3$ (Sobolev loss)	**6.49e**−04 (**3.03e**−04)	**1.45e**−02 (**6.35e**−03)	**7.28e**−02 (**3.80e**−02)	**2.64e**−01 (**1.42e**−01)	**5.74e**−01 (**2.78e**−01)	**8.83e**−01 (**3.21e**−01)

Table 3: **Comparison of rollout MSEs with varied training objectives**. We train the models using (1) standard $\ell_1 + \ell_2$ loss functions; (2) Sobolev loss from Li et al. [25].

Method	Seconds per iter.	1-step MSE	25-step MSE	50-step MSE	Zonal mean error
Baseline: $L = 1$	0.2067	5.60e−04 (1.15e−04)	8.04e−02 (3.45e−02)	3.12e−01 (1.15e−01)	4.20 (2.36)
Pushforward [18]	0.2834	1.08e−03 (2.26e−04)	9.16e−02 (4.62e−02)	3.19e−01 (1.34e−01)	1.07 (0.44)
Ours: $L = 3$	0.2204	**5.50e-04 (1.20e-04)**	**3.37e-02(1.41e-02)**	**1.40e-01 (6.16e-02)**	**0.63 (0.58)**

Table 4: **Comparison of training efficiency and multi-step forecasting errors.** Our $L = 3$ method yields better short-term accuracy and preserves better structure coherence, compared to the $L = 1$ baseline and the pushforward unrolling [18].

	512×512 resolution, w jet			256×256 resolution w/o jet		
Rollout Step	25	50	75	25	50	75
Baseline: $L = 1$	2.16e-01 (8.77e-02)	6.82e-01 (2.55e-01)	1.13e+00 (3.54e-01)	2.59e-01 (1.09e-01)	8.44e-01 (2.88e-01)	1.42e+00 (3.92e-01)
Ours: $L = 3$	**5.75e-02 (3.04e-02)**	**1.81e-01 (9.02e-02)**	**3.89e-01 (1.83e-01)**	**1.28e-01 (5.13e-02)**	**5.12e-01 (1.76e-01)**	**9.98e-01 (2.55e-01)**

Table 5: **Mean squared error on different datasets for short-term rollout.** We report mean (standard deviation) across 100 trials of short-term rollout with different initial conditions.

Ablation on training objectives. In addition to the standard $l1$ and $l2$ losses, we further provide results for experiments with the Sobolev loss, proposed in Li et al. [25]. As shown in Table 3, across different training objectives, our $L = 3$ method consistently yields significant improvements compared to the $L = 1$ baseline.

Comparison on training efficiency and rollout method. Unlike multi-step unrolling methods [18, 20], our approach achieves greater computational efficiency while enhancing both short-term accuracy and long-term coherence. As illustrated in Table 4, it incurs only about a 10% higher cost per iteration relative to the $L = 1$ baseline, yet substantially reduces errors across both short- and long-horizon metrics. In contrast, the pushforward method [18] mitigates distributional shift at the expense of higher computational cost and degraded short-term accuracy.

5.3 Performance on Diverse Datasets

We further evaluate the robustness of our approach across multiple datasets and spatial resolutions. In addition to the jet-forming regime with $\beta = 20$ that mimics the influence of Earth's rotation, we consider an eddy configuration ($\beta = 0$). As shown in Fig. 6, our approach ($L = 3$) consistently produces realistic and accurate flow evolutions. In contrast, the baseline model ($L = 1$) shows large residuals after a short rollout. This performance gap is further quantified in Table 5, where the MSE of $L = 1$ grows rapidly over time, while $L = 3$ maintains low and stable errors.

6 Conclusion

In this work, we present a novel framework for autoregressive modeling of dynamical systems, inspired by implicit time-stepping numerical solvers. Diverging from prior approaches, our method uniquely integrates future-state insights to refine next-step predictions. To enable autoregressive operation, we introduce a temporal hierarchy via spatial compression, coupled with an encoder-decoder architecture. This design organizes multiscale spatial-temporal information, empowering the encoder to distill critical features and the decoder to adaptively address prediction challenges across scales. Evaluated on turbulent flow systems, our framework achieves substantial error reduction while maintaining computational efficiency. While our current focus centers on deterministic modeling, the architecture's flexibility suggests promising extensions to generative paradigms (e.g., diffusion models) for probabilistic forecasting and uncertainty quantification.

Broader Impacts. While better emulators for dynamical systems could be used in a wide range of applications, we foresee no direct negative societal impacts.

Acknowledgements

RJ, PL, and RW gratefully acknowledge the support of AFOSR FA9550-18-1-0166, the Eric and Wendy Schmidt AI in Science Fellowship program, and the Margot and Tom Pritzker Foundation. RJ, PL, RW, and MM gratefully acknowledge the support of the NSF-Simons AI-Institute for the Sky (SkAI) via grants NSF AST-2421845 and Simons Foundation MPS-AI-00010513. KJ and PH were supported by NSF RISE-2425898 and Schmidt Sciences, LLC.

References

[1] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[2] Ruoxi Jiang, Peter Y. Lu, and Rebecca Willett. Embed and emulate: Contrastive representations for simulation-based inference. *arXiv preprint arXiv:2409.18402*, 2024.

[3] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[4] Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, et al. Neural general circulation models for weather and climate. *Nature*, 632(8027):1060–1066, 2024.

[5] Oliver Watt-Meyer, Gideon Dresdner, Jeremy McGibbon, Spencer K Clark, Brian Henn, James Duncan, Noah D Brenowitz, Karthik Kashinath, Michael S Pritchard, Boris Bonev, et al. Ace: A fast, skillful learned global atmospheric model for climate prediction. *arXiv preprint arXiv:2310.02074*, 2023.

[6] Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, Feb 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-36329-y.

[7] Raidel Martin-Barrios, Edisel Navas-Conyedo, Xuyi Zhang, Yunwei Chen, and Jorge Gulín-González. An overview about neural networks potentials in molecular dynamics simulation. *International Journal of Quantum Chemistry*, 124(11):e27389, 2024.

[8] Raimondas Galvelis and Yuji Sugita. Neural network and nearest neighbor algorithms for enhancing sampling of molecular dynamics. *Journal of chemical theory and computation*, 13 (6):2489–2500, 2017.

[9] Drew Jamieson, Yin Li, Renan Alves de Oliveira, Francisco Villaescusa-Navarro, Shirley Ho, and David N Spergel. Field-level neural network emulator for cosmological n-body simulations. *The Astrophysical Journal*, 952(2):145, 2023.

[10] Drew Jamieson, Yin Li, Siyu He, Francisco Villaescusa-Navarro, Shirley Ho, Renan Alves de Oliveira, and David N Spergel. Simple lessons from complex learning: what a neural network model learns about cosmic structure formation. *PNAS nexus*, 2(4):pgac250, 2023.

[11] Siyu He, Yin Li, Yu Feng, Shirley Ho, Siamak Ravanbakhsh, Wei Chen, and Barnabás Póczos. Learning to predict the cosmological structure formation. *Proceedings of the National Academy of Sciences*, 116(28):13825–13832, 2019.

[12] Jeongjin Park, Nicole Yang, and Nisha Chandramoorthy. When are dynamical systems learned from time series data statistically accurate? *arXiv preprint arXiv:2411.06311*, 2024.

[13] Massimo Bonavita. On some limitations of current machine learning weather prediction models. *Geophysical Research Letters*, 51(12):e2023GL107377, 2024.

[14] Ashesh Chattopadhyay, Y. Qiang Sun, and Pedram Hassanzadeh. Challenges of learning multi-scale dynamics with ai weather models: Implications for stability and one solution, 2024. URL https://arxiv.org/abs/2304.07029.

[15] Ching-Yao Lai, Pedram Hassanzadeh, Aditi Sheshadri, Maike Sonnewald, Raffaele Ferrari, and Venkatramani Balaji. Machine learning for climate physics and simulations. *Annual Review of Condensed Matter Physics*, 16, 2024.

[16] Niraj Agarwal, Dmitri Kondrashov, Peter Dueben, Eugene Ryzhov, and Pavel Berloff. A comparison of data-driven approaches to build low-dimensional ocean models. *Journal of Advances in Modeling Earth Systems*, 13(9):e2021MS002537, 2021.

[17] Jayesh K. Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling, 2022. URL https://arxiv.org/abs/2209.15616.

[18] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

[19] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning, 2024. URL https://arxiv.org/abs/2210.07182.

[20] Bjoern List, Li-Wei Chen, Kartik Bali, and Nils Thuerey. Differentiability in unrolled training of neural physics simulators on transient dynamics. *Computer Methods in Applied Mechanics and Engineering*, 433:117441, 2025.

[21] Salva Rühling Cachay, Bo Zhao, Hailey Joren, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *Advances in neural information processing systems*, 36:45259–45287, 2023.

[22] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[23] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator: a neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022.

[24] Peiyan Hu, Rui Wang, Xiang Zheng, Tao Zhang, Haodong Feng, Ruiqi Feng, Long Wei, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Wavelet diffusion neural operator. *arXiv preprint arXiv:2412.04833*, 2024.

[25] Zongyi Li, Miguel Liu-Schiaffini, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Learning chaotic dynamics in dissipative systems. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=1C36tFZn7sR.

[26] Ruoxi Jiang, Peter Y. Lu, Elena Orlova, and Rebecca Willett. Training neural operators to preserve invariant measures of chaotic attractors. *Advances in Neural Information Processing Systems*, 36, 2024.

[27] Yinnian He. Unconditional convergence of the euler semi-implicit scheme for the three-dimensional incompressible mhd equations. *IMA Journal of Numerical Analysis*, 35(2):767–801, 2015.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[29] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[30] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *IEEE/CVF International Conference on Computer Vision*, 2021.

[31] Michael Maire and Stella X Yu. Progressive multigrid eigensolvers for multiscale spectral segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2184–2191, 2013.

[32] Tsung-Wei Ke, Michael Maire, and Stella X Yu. Multigrid neural architectures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6665–6673, 2017.

[33] Tri Huynh, Michael Maire, and Matthew R. Walter. Multigrid neural memory. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[34] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

[35] Xiao Zhang, Ruoxi Jiang, Rebecca Willett, and Michael Maire. Nested diffusion models using hierarchical latent priors. *arXiv preprint arXiv:2412.05984*, 2024.

[36] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*, 2023.

[37] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[38] John W Ruge and Klaus Stüben. Algebraic multigrid. In *Multigrid methods*, pages 73–130. SIAM, 1987.

[39] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[40] Jie Chen, Yousef Saad, and Zechen Zhang. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, 79(1):187–223, 2022.

[41] Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

[42] Qi Wang, Yuan Mi, Haoyun Wang, Yi Zhang, Ruizhi Chengze, Hongsheng Liu, Ji-Rong Wen, and Hao Sun. Multipdenet: Pde-embedded learning with multi-time-stepping for accelerated flow simulation. *arXiv preprint arXiv:2501.15987*, 2025.

[43] Uri M Ascher, Steven J Ruuth, and Brian TR Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

[44] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM review*, 43(1):89–112, 2001.

[45] Jason Frank, Willem Hundsdorfer, and Jan G Verwer. On the stability of implicit-explicit linear multistep methods. *Applied Numerical Mathematics*, 25(2-3):193–205, 1997.

[46] Eugene Isaacson and Herbert Bishop Keller. *Analysis of numerical methods*. Courier Corporation, 2012.

[47] Karan Jakhar, Yifei Guan, Rambod Mojgani, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Learning closed-form equations for subgrid-scale closures from high-fidelity data: Promises and challenges. *Journal of Advances in Modeling Earth Systems*, 16(7):e2023MS003874, 2024.

[48] Yifei Guan, Ashesh Chattopadhyay, Adam Subel, and Pedram Hassanzadeh. Stable a posteriori les of 2d turbulence using convolutional neural networks: Backscattering analysis and generalization to higher re via transfer learning. *Journal of Computational Physics*, 458:111090, 2022.

[49] Chris Pedersen, Laure Zanna, and Joan Bruna. Thermalizer: Stable autoregressive neural emulation of spatiotemporal chaos. *arXiv preprint arXiv:2503.18731*, 2025.

[50] Romit Maulik, Omer San, Adil Rasheed, and Prakash Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, 2019.

[51] Geoffrey K Vallis. *Atmospheric and oceanic fluid dynamics*. Cambridge University Press, 2017.

[52] Karan Jakhar, Rambod Mojgani, Moein Darman, Yifei Guan, and Pedram Hassanzadeh. py2d: High-performance 2D Navier-Stokes solver in python (version 0.1), 2024. URL https://github.com/envfluids/py2d.

[53] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

[54] Stephen B Pope. Turbulent flows. *Measurement Science and Technology*, 12(11):2020–2021, 2001.

[55] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: In the abstract, we described our goal as building a powerful neural emulator.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitation in the conclusion.

Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

   Justification: We don't provide theoretical results.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We describe our detailed setup in our experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

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

5. **Open access to data and code**

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

   Answer: [No] for the code.

   Justification: We have disclosed details of experimental setups for reproducing the results. Our code will be released upon acceptance.

   Guidelines:

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

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide details in the experiment section.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report std in Figure 3, 4 Table 2, 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We describe those details in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: None

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Yes, we provide the Broader impacts in the paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: Our work doesn't introduce any security concerns.

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [NA]

    Justification: We don't use existing assets.

    Guidelines:

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

13. **New assets**

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

    Answer: [NA]

    Justification: paper does not release new assets

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

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

Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

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

    Guidelines:

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

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 rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA 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 (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Experimental details.

A.1 Further experiments.

Rollout steps	1	25	50	75	100
Baseline : $L = 1$	5.60e-04 (1.15e-04)	8.04e-02 (3.45e-02)	3.12e-01 (1.15e-01)	6.19e-01 (2.08e-01)	9.38e-01 (2.88e-01)
FNO [22] (4x Params.)	1.58e-03 (2.82e-04)	5.22e-02 (3.12e-02)	1.94e-01 (9.78e-02)	4.22e-01 (1.95e-01)	7.12e-01 (2.24e-01)
Spatial Hierarchical	5.15e-04 (1.14e-04)	6.59e-02 (3.59e-02)	2.69e-01 (1.31e-01)	5.77e-01 (2.39e-01)	9.28e-01 (3.08e-01)
History Hierarchy	5.11e-04 (1.14e-04)	6.41e-02 (4.03e-02)	2.36e-01 (1.23e-01)	4.99e-01 (2.29e-01)	7.79e-01 (2.70e-01)
2-step Ahead	1.06e-03 (2.18e-04)	4.22e-02 (1.87e-02)	1.79e-01 (7.70e-02)	4.11e-01 (1.69e-01)	7.17e-01 (2.40e-01)
2-step History [19]	4.84e-04 (1.09e-04)	5.21e-02 (2.76e-02)	2.02e-01 (8.99e-02)	4.42e-01 (1.86e-01)	7.67e-01 (2.71e-01)
3-step History [19]	**4.53e-04 (9.65e-05)**	4.28e-02 (2.10e-02)	1.81e-01 (8.71e-02)	3.96e-01 (1.57e-01)	6.81e-01 (2.39e-01)
Ours: $L = 2$	5.25e-04 (1.15e-04)	4.02e-02 (1.92e-02)	1.61e-01 (6.50e-02)	3.73e-01 (1.51e-01)	6.55e-01 (2.44e-01)
Ours: $L = 3$	5.50e-04 (1.20e-04)	**3.37e-02 (1.41e-02)**	**1.40e-01 (6.16e-02)**	**3.24e-01 (1.18e-01)**	**5.92e-01 (1.84e-01)**

Table 6: **Comparisons to other methods.** We compare to other methods using mean squared error (MSE) for autoregressive roll-out across different lengths. We report the results using average and standard deviation, in parentheses, and demonstrate that ours ($L = 3$) outperforms others in all steps above 1-step MSE. We also show that building models with both spatial and temporal hierarchies enhances the stability of the estimation.

Rollout steps	1	25	50	75	100
Baseline: L=1	8.02e-07 (4.89e-07)	9.25e-06 (4.33e-06)	1.49e-05 (5.56e-06)	2.16e-05 (7.40e-06)	2.93e-05 (1.05e-05)
FNO [22] (4x Params.)	**4.90e-07 (1.91e-07)**	4.93e-06 (1.61e-06)	9.68e-06 (2.59e-06)	1.52e-05 (3.71e-06)	2.14e-05 (4.97e-06)
Spatial-Hierarchy	6.28e-07 (3.96e-07)	1.03e-05 (4.83e-06)	1.87e-05 (7.54e-06)	2.65e-05 (9.94e-06)	3.48e-05 (1.14e-05)
History-Hierarchy	8.52e-07 (6.56e-07)	8.95e-06 (5.34e-06)	1.33e-05 (6.66e-06)	1.84e-05 (8.33e-06)	2.42e-05 (9.50e-06)
2-step Ahead	5.89e-07 (4.10e-07)	**5.88e-06 (3.29e-06)**	9.94e-06 (4.42e-06)	1.49e-05 (5.71e-06)	2.10e-05 (7.75e-06)
2-step History [19]	9.83e-07 (6.22e-07)	8.11e-06 (4.29e-06)	1.13e-05 (4.90e-06)	1.58e-05 (5.87e-06)	2.18e-05 (7.06e-06)
3-step History [19]	5.65e-07 (4.40e-07)	7.05e-06 (3.93e-06)	1.09e-05 (5.08e-06)	1.55e-05 (6.30e-06)	2.12e-05 (8.01e-06)
Ours: L=2	7.86e-07 (4.75e-07)	9.42e-06 (4.91e-06)	1.24e-05 (5.59e-06)	1.62e-05 (6.14e-06)	2.10e-05 (6.24e-06)
Ours: L=3	6.37e-07 (3.80e-07)	6.28e-06 (2.90e-06)	**9.29e-06 (3.72e-06)**	**1.28e-05 (4.52e-06)**	**1.76e-05 (5.44e-06)**

Table 7: **Energy spectrum error comparison.** Our $L = 3$ model achieves significant reduction in energy spectrum error during short-term rollouts (up to 200 steps), outperforming all comparison methods - including a baseline with $4\times$ more parameters - for all prediction horizons beyond single-step forecasting.

Experimenting with FNO. All experiments thus far have utilized a UNet architecture with Fourier layers, as described in Section 5. To provide additional comparison, we run $L = 1$ with Fourier Neural Operator (FNO) [55]. Key adaptations include: (1) Using the $\ell_1 + \ell_2$ loss (consistent with our other experiments in Table 6) instead of the default Sobolev-norm objective [25], as the latter caused rapid prediction divergence (even within 200 steps); (2) For 256×256 resolution data, we choose the Fourier mode number through grid search over $\{64, 96, 128\}$; (3) We use 3 FNO layers, which results in a model with 130M parameters — $4\times$ larger than our $L = 3$ configuration.

Model Efficiency. Our model utilizes the UNet's hierarchical framework to efficiently process multi-scale data. As shown in Table 8, compared to the baseline $L = 1$, ours $L = 3$ adds only a few convolutional heads for handling and outputting latent variables $\boldsymbol{z}$, resulting in minimal parameter and inference time overhead.

	Param Counts (M)	Forward Time (seconds)
Baseline: $L = 1$	34.88	0.030
Ours: $L = 3$	36.67	0.031

Table 8: **Model Running Time and Parameters comparison**. On input with 256×256 resolution, our design ($L = 3$) leverages the inherent hierarchical representation of UNet to process hierarchical latent variables, leading to minimal computational overhead than the baseline ($L = 1$).

A.2 Evaluation metrics.

Mean squared error (MSE). We use

$$\mathrm{MSE}(\boldsymbol{u}_n, \hat{\boldsymbol{u}}_n) = \|\boldsymbol{u}_n - \hat{\boldsymbol{u}}_n\|_2^2$$

as our primary metric to quantify the prediction error for short-term predictions.

Long-term stability. To evaluate the long-term stability of predictions, we leverage the system's conserved energy, defined as $E = \frac{1}{2}(v_x^2 + v_y^2)$. As discussed in Section 5.1, we calculate the standard deviation of the ground truth energy values. A trajectory is deemed stable if its energy remains within 5 standard deviations of this reference. This threshold is informed by the observation that all training data lie within 4 standard deviations, while even the true dynamics extrapolated over ten times the training horizon (simulating extreme long-term behavior) do not exceed 5 standard deviations, as shown in Fig. 7. Appendix B.2 provides a detailed visual comparison when the predicted dynamics exceed ± 5 standard deviation range.

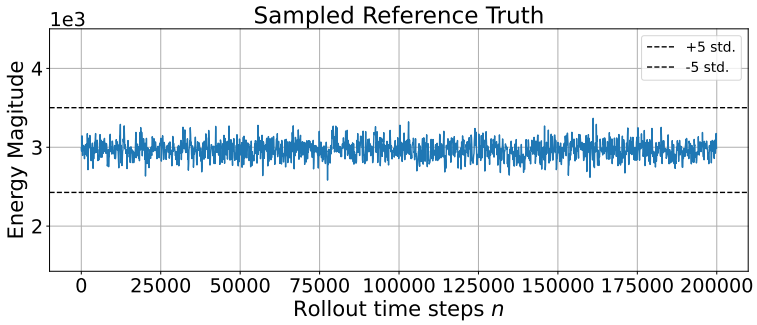


Figure 7: **Energy of true dynamics along extended timescale.** We compute the energy evolution of the true dynamics over 10-times the training dataset length (2×10^5 time steps). We show that energy along the sampled true dynamics remains within the ± 5 standard deviation of its mean.

Fourier spectrum error. We follow the implementation in py2d to compute the energy spectrum[2]. For short-term evaluation, we compute the mean absolute error of the energy spectrum—the spatial FFT $\mathcal{F}[\boldsymbol{u}_n]$, averaged over N timesteps:

$$\frac{1}{N}\sum_{n=1}^{N} \|\mathcal{F}[\boldsymbol{u}_n] - \mathcal{F}[\hat{\boldsymbol{u}}_n]\|_1.$$

For long-term evaluation (e.g., $N = 2 \times 10^5$), where a ground truth trajectory is unavailable for every new initial condition, and the system loses memory of initial conditions showing an invariant spectrum due to its ergodic properties, we instead compare against a fixed reference spectrum $\mu[\mathcal{F}[\boldsymbol{u}]] := \frac{1}{N}\sum_{n=1}^{N} \mathcal{F}[\boldsymbol{u}_n]$, computed from an extremely long true trajectory. The error is then:

$$\left\| \frac{1}{N}\sum_{n=1}^{N} \mathcal{F}[\boldsymbol{u}_n] - \mu[\mathcal{F}[\boldsymbol{u}]] \right\|_1.$$

Zonal mean. The zonal mean of vorticity is computed by averaging vorticity perpendicular to the jets (i.e., along the x-direction), shown as:

$$\frac{1}{NN_x}\sum_{n=1}^{N}\sum_{x=1}^{N_x} \boldsymbol{u}_{n,x}.$$

$\boldsymbol{u}_{n,x} \in \mathbb{R}^d$ denotes the vector of the vorticity at x-axis. N_x denotes the number of discretization points.

[2]https://github.com/envfluids/py2d/blob/main/py2d/spectra.py

A.3 Experimental setup.

Model Architecture We adopt the UNet architecture following the design in [53], with several customizations. For the experiments on 256×256 resolution data, our encoder consists of 5 groups with latent channel sizes of $[16, 32, 64, 128, 128, 128]$. The spatial resolution is halved after each encoder group, resulting in a 8×8 resolution at the bottleneck for 256×256 input images. Each encoder group contains two convolutional blocks, which is made up of two convolutional layers with group normalization and residual connections. The decoder mirrors the encoder and incorporates skip connections from corresponding encoder layers. To improve the model's ability to capture long-range spatial dependencies and multi-frequency signals, we add a Fourier layer to each convolutional block. To balance between accuracy and computational efficiency, we limit the number of frequency modes to 96 for 256×256 data and apply Fourier convolutions in a depth-wise manner, *i.e.*, without inter-channel communication.

For 512×512 resolution inputs, we use a similar architecture but add an extra encoder group, resulting in latent channel sizes of $[16, 32, 32, 64, 128, 128, 128]$. In the $L = 3$ setup, we set $r^1 = 8$ and $r^2 = 32$, the same rates used for our 256×256 resolution experiments. Using these rates, we inject latent codes $\boldsymbol{z}_1^{(1)}$ and $\boldsymbol{z}_2^{(2)}$ into encoder groups with feature resolutions of 64×64 and 32×32.

Data preprocessing. To stabilize training, we normalize the data using a constant scaling factor such that the resulting values have an approximate standard deviation of one. Specifically, for the dataset with Reynolds number 10000, we apply a dividing factor of 10, while for the dataset with Reynolds number 5000, we use a dividing factor of 6.

Corner cases. For models that take multiple temporal frames as input, we simulate the initial rollout setting during training by randomly zeroing out early history frames with a probability of 15%, approximating the absence of pre-initial frames.

Upsampling projections. Prior to injection, we upsample the latent variables and apply convolutional layers to match the corresponding encoder channels. Decoding is performed at the same spatial resolutions as the injected latent codes. The same architectural setup as $L = 3$ is used to construct baseline models for Spatial Hierarchy and History Hierarchy. In the $L = 2$ case, we use $r^1 = 32$ and inject $\boldsymbol{z}_1^{(1)}$ into the encoder group at the 64×64 resolution level.

Training details. We process the raw data generated from "py2d" [52] solver. Our emulator predicts vorticity at 0.05 time intervals, representing a $500\times$ coarser temporal resolution than the numerical solver's 0.0001 timestep. We normalize the raw simulation data to approximate a standard normal distribution. Since the data is naturally zero-centered, we simply scale it by constant factors (10 for jet-containing flows and 6 for jet-free flows in Section 5.3) to achieve unit standard deviation.

For all experiments, we use the AdamW optimizer with learning rate at 3×10^{-4}. We conduct experiments on NVIDIA A100, H100, and L40S GPUs.

B Additional visualizations.

B.1 Further visualizations.

We provide additional visualization of short-term rollout in Fig. 8, Fig. 9, Fig. 10, and Fig. 11. The comparison across various datasets and methods show consistent improvements of our methods ($L = 2$ and $L = 3$).

B.2 Stability.

We present four trials of long-term rollouts across all methods, each with distinct initial conditions. The energy evolution over 2×10^5 timesteps is shown in Figures 12 and 15, while the corresponding dynamic visualizations appear in Figures 13 through 17.

Our analysis reveals that deviations beyond the ± 5 standard deviations range of the ground truth energy distribution consistently correlate with unstable, exploding dynamics or overly smoothed, averaged patterns. While baseline methods exhibit persistent failures once instability occurs, our $L = 2$ model demonstrates a unique recovery pattern, where the dynamics (and associated energy values) can return to physically plausible states after temporary deviations. This robustness stems

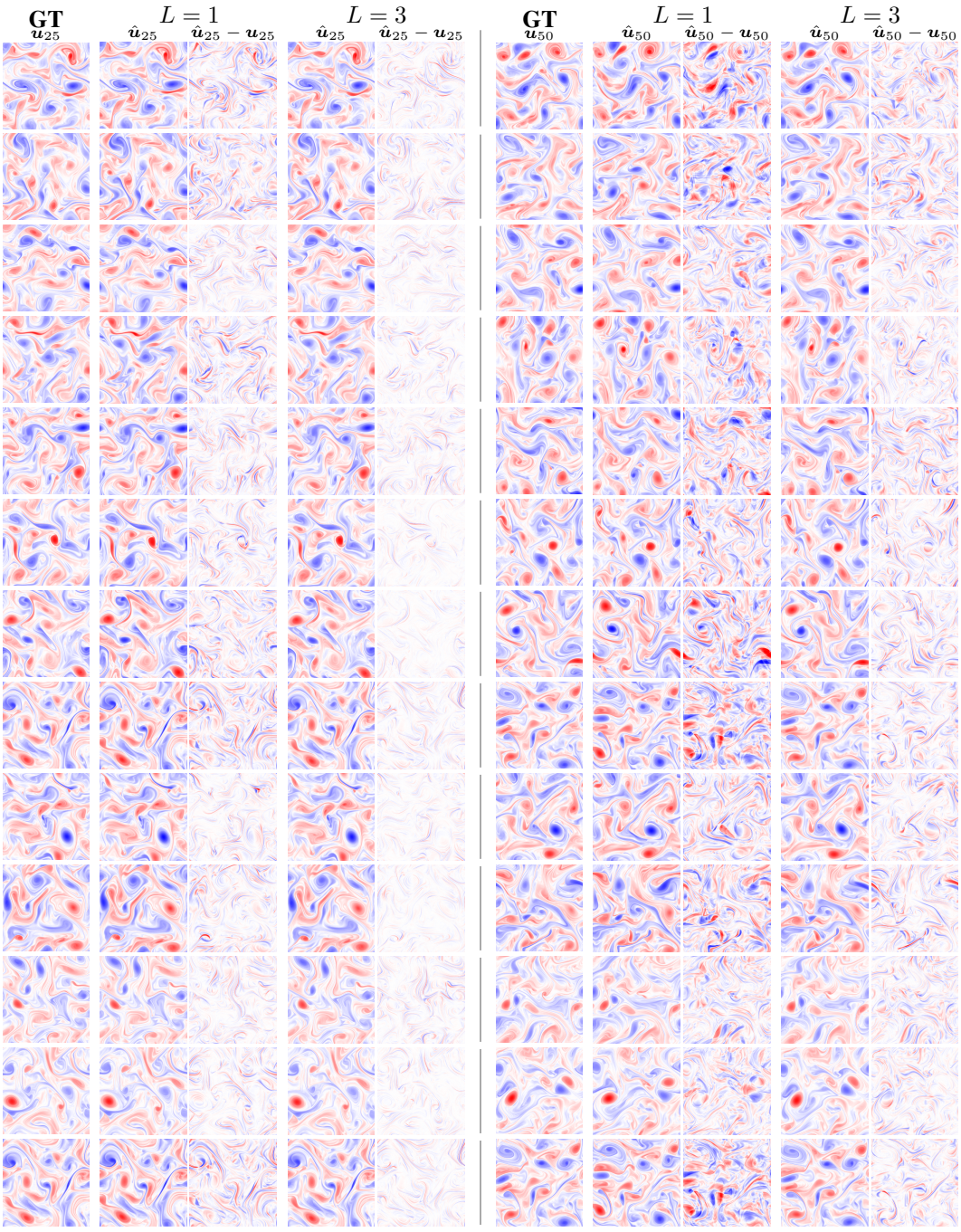


Figure 8: We apply our approach to flow dataset of $Re = 5 \times 10^3$, 256×256 resolution without zonal jets Our method ($L = 3$) gives more accurate predictions with lower associated residuals.

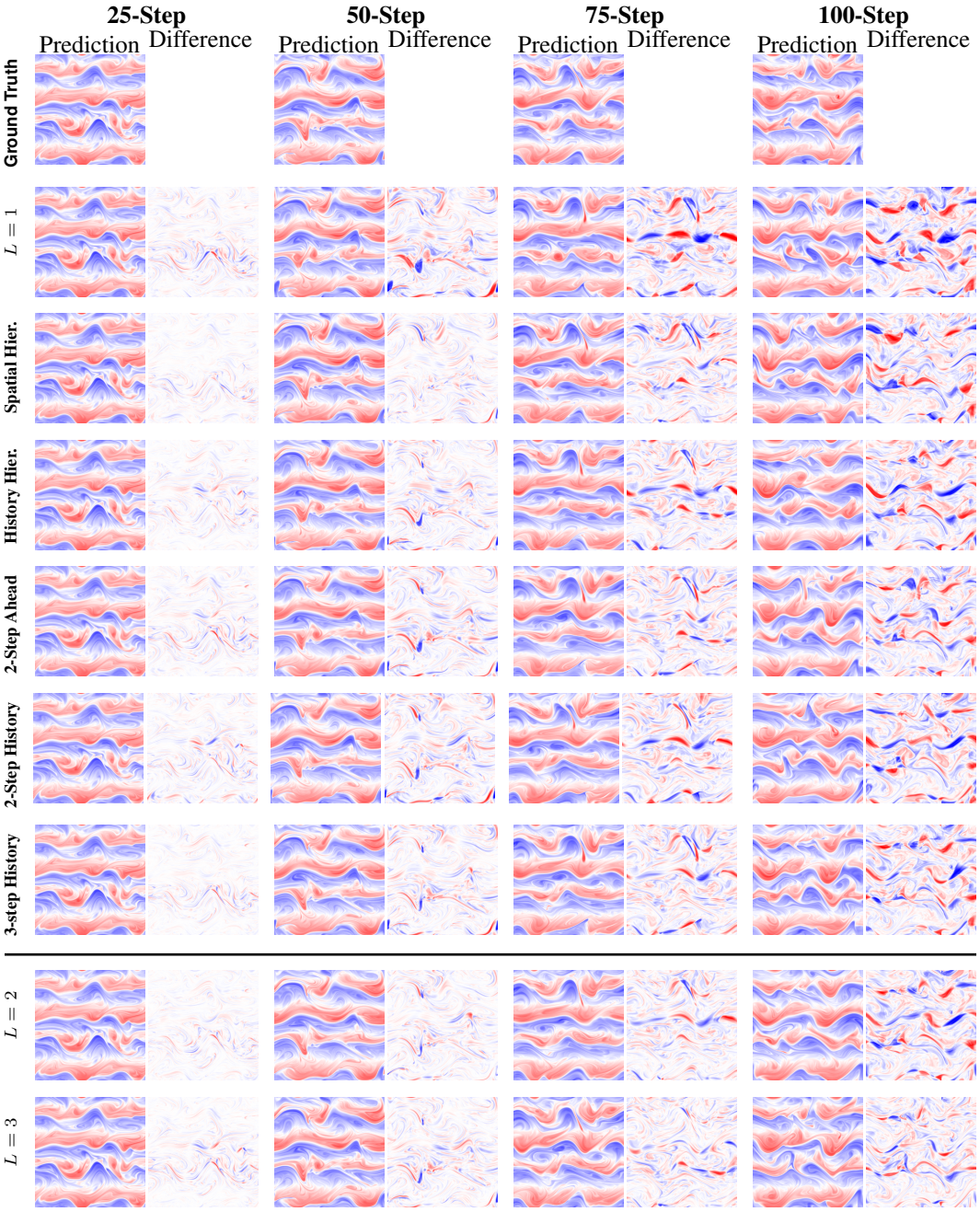


Figure 9: Visualization of prediction and residual to ground truth across methods and prediction steps. Our methods $L = 2, L = 3$ consistently outperform all compared methods.

from the guidance provided by the top-level compressed variables, which help correct errors in fine-grained details.

For stability rate calculations, we conservatively classify a trajectory as unstable if it ever exceeds the predefined ± 5 standard deviations' bounds, regardless of subsequent recovery. This ensures fair comparison across methods.

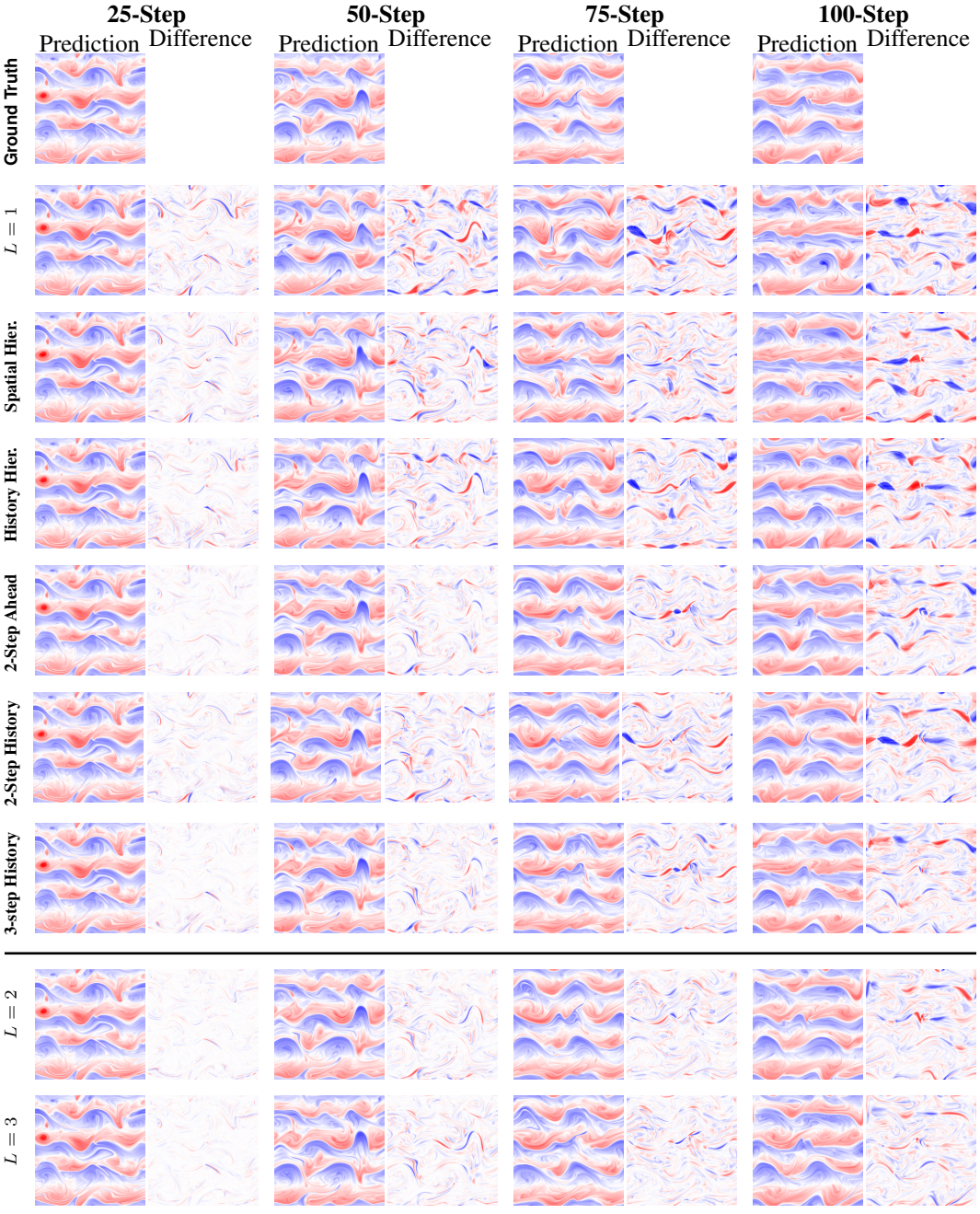


Figure 10: Visualization of prediction and residual to ground truth across methods and prediction steps. Our methods $L = 2, L = 3$ consistently outperform all compared methods.

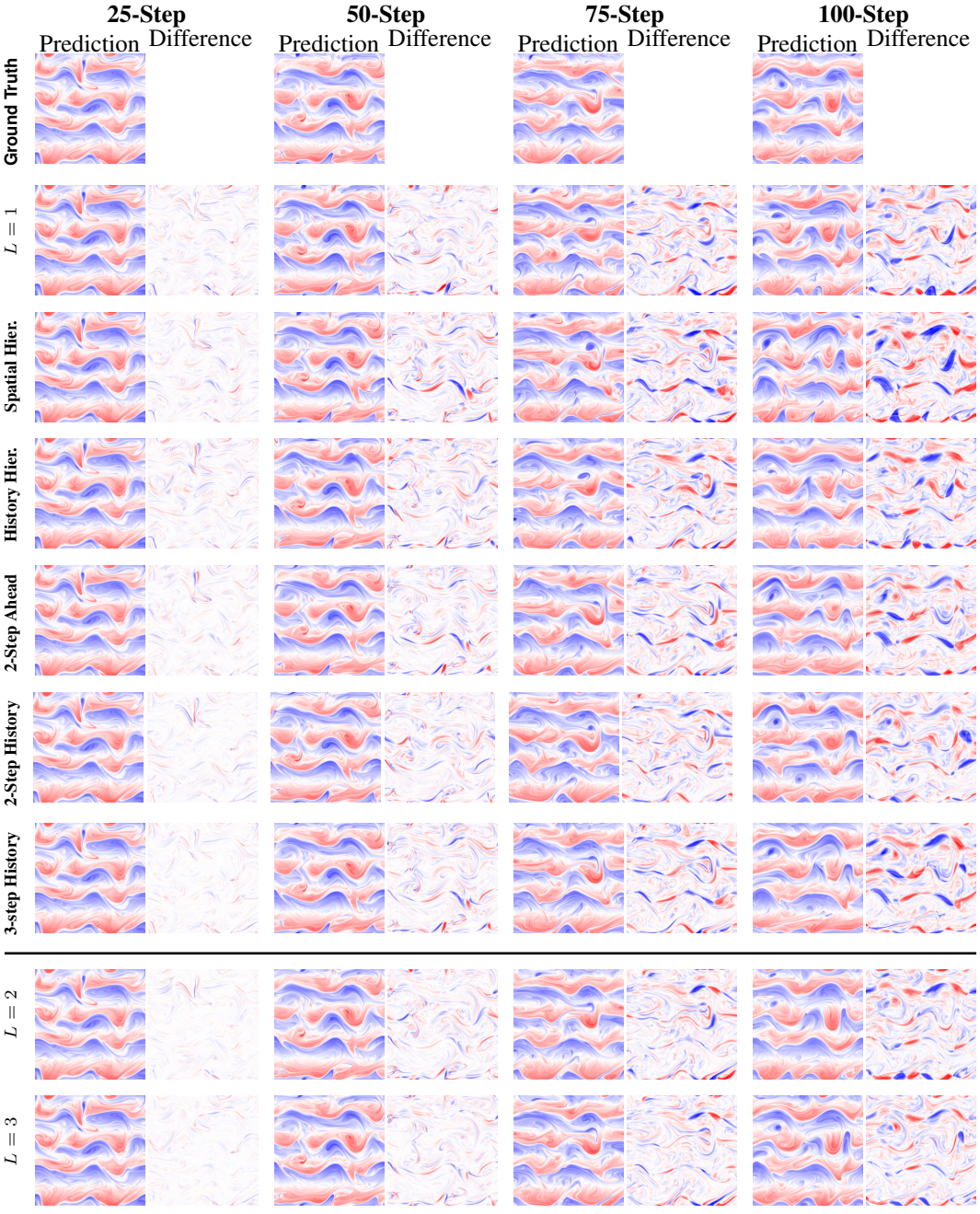


Figure 11: Visualization of prediction and residual to ground truth across methods and prediction steps. Our methods $L = 2, L = 3$ consistently outperform all compared methods.

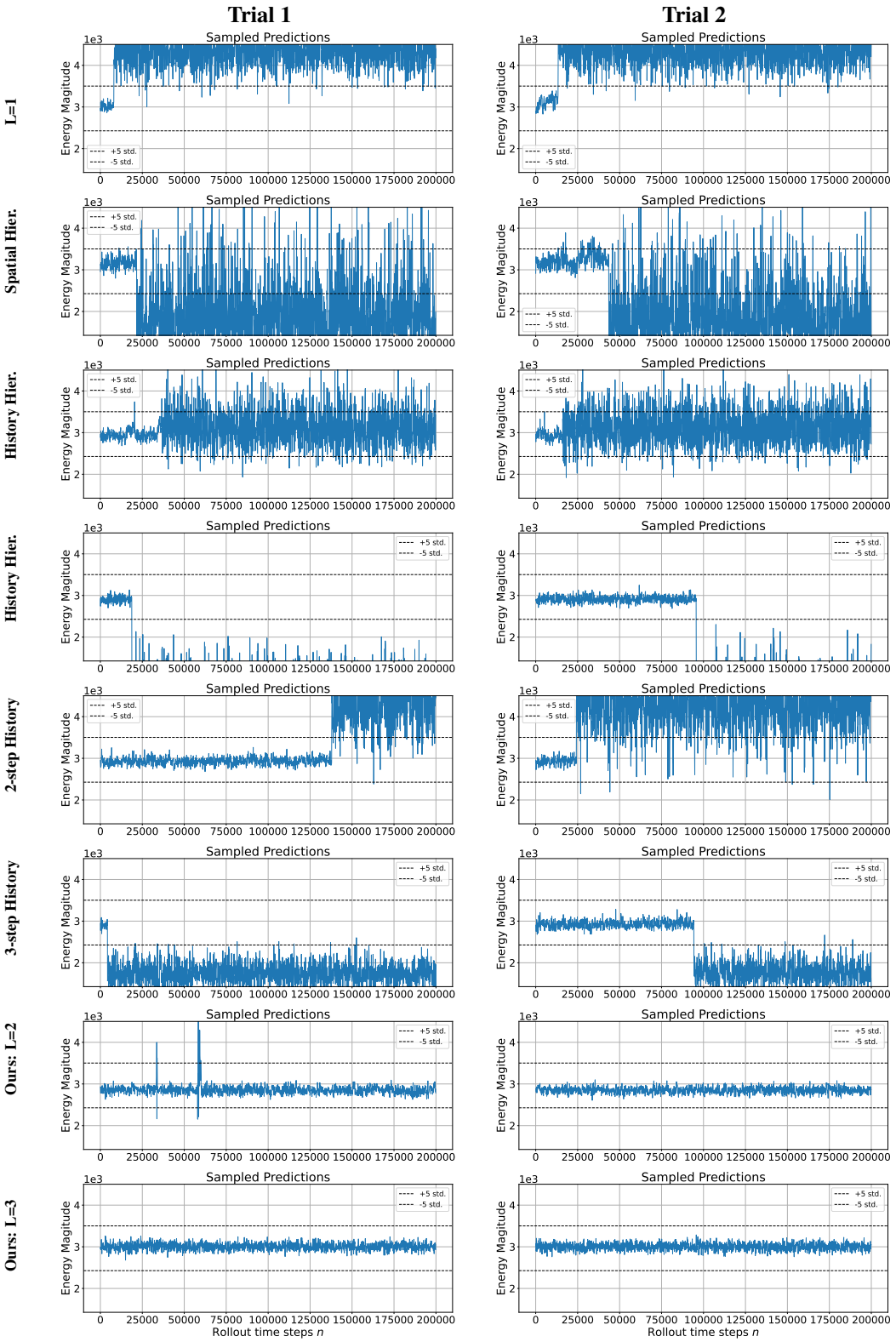


Figure 12: **Energy v.s. rollout time steps of the predicted dynamics.** The dashed horizontal line shows the maximum and minimum values computed using 5 standard deviations from the mean of the true dynamics. Our $L = 3$ method is able to maintain energy along the long-term predictions.

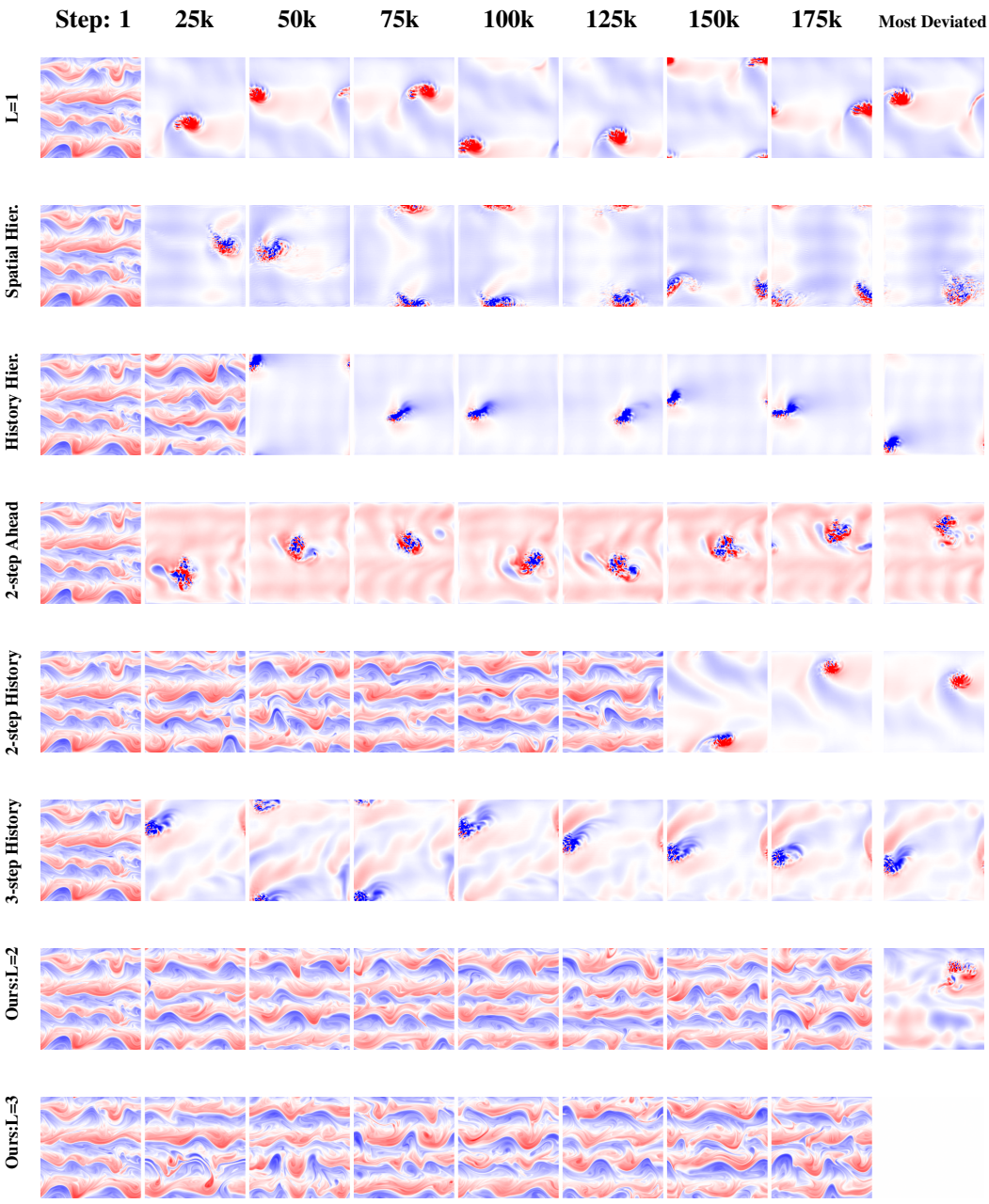


Figure 13: **Long-Term Rollout Visualization.** Long-term dynamics corresponding to Trial 1 in Fig. 12. The rightmost column shows the frame with the maximum deviation—exceeding ± 5 standard deviations from the reference truth statistics, with a blank block indicating that all states along the sampled trajectory remain within this range. When energy predictions fall outside the reference distribution, non-physical dynamics emerge (exploding/averaging artifacts; blank blocks indicate predictions within the ± 5 std. range). Our $L = 3$ model demonstrates superior stability across all comparisons, while $L = 2$ exhibits deviations correlated with energy prediction errors.

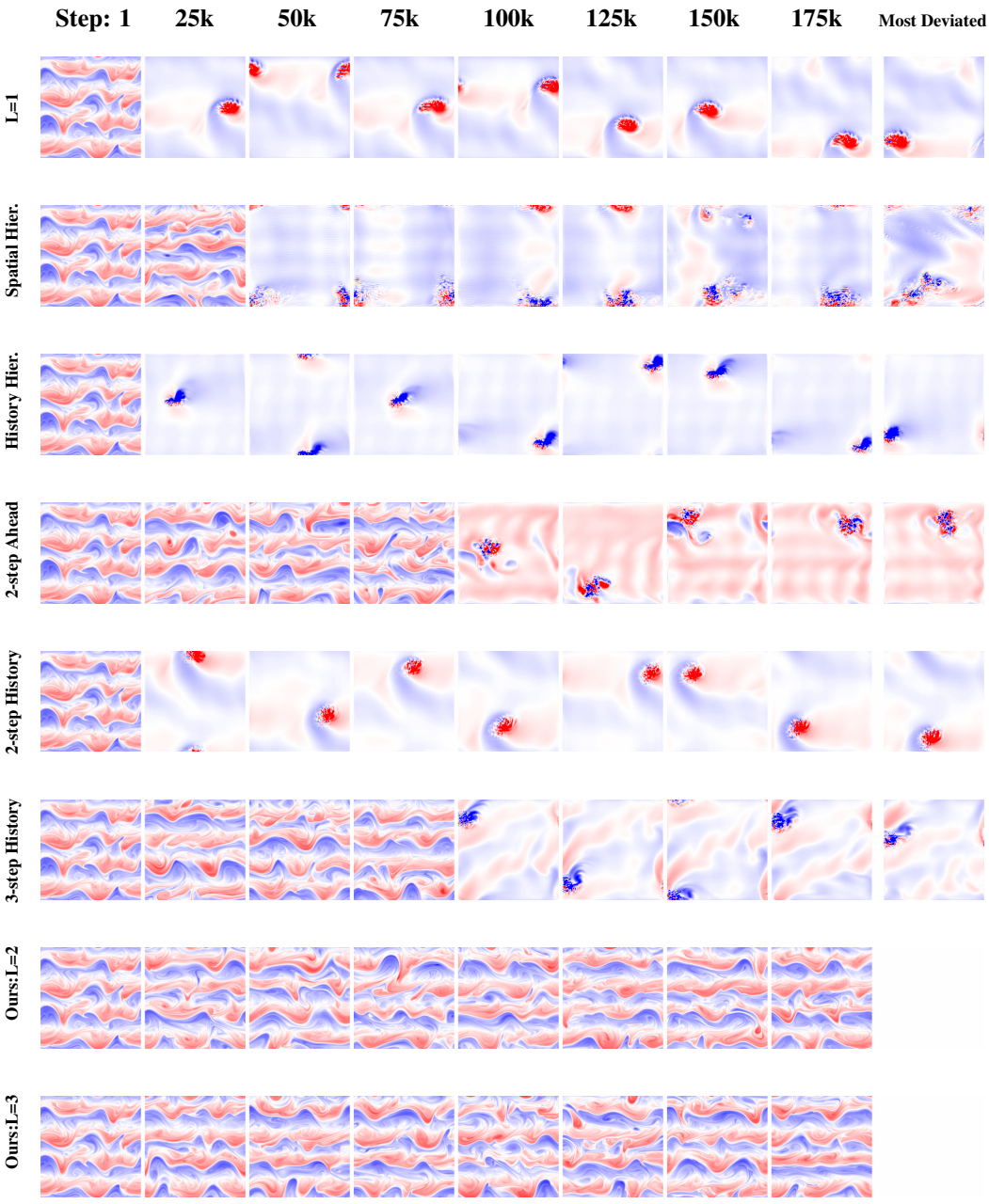


Figure 14: **Visualization of long-term rollout.** We visualize the long-term dynamics corresponds to trial 2 in Fig. 12. The rightmost column shows the frame with the maximum deviation—exceeding ± 5 standard deviations from the reference truth statistics, with a blank block indicating that all states along the sampled trajectory remain within this range. The results show that when the energy is deviated from the reference truth distribution, the dynamics exhibit exploding or averaging non-physical dynamics. Among all comparison methods, our $L = 3$ gives the most stable predictions.

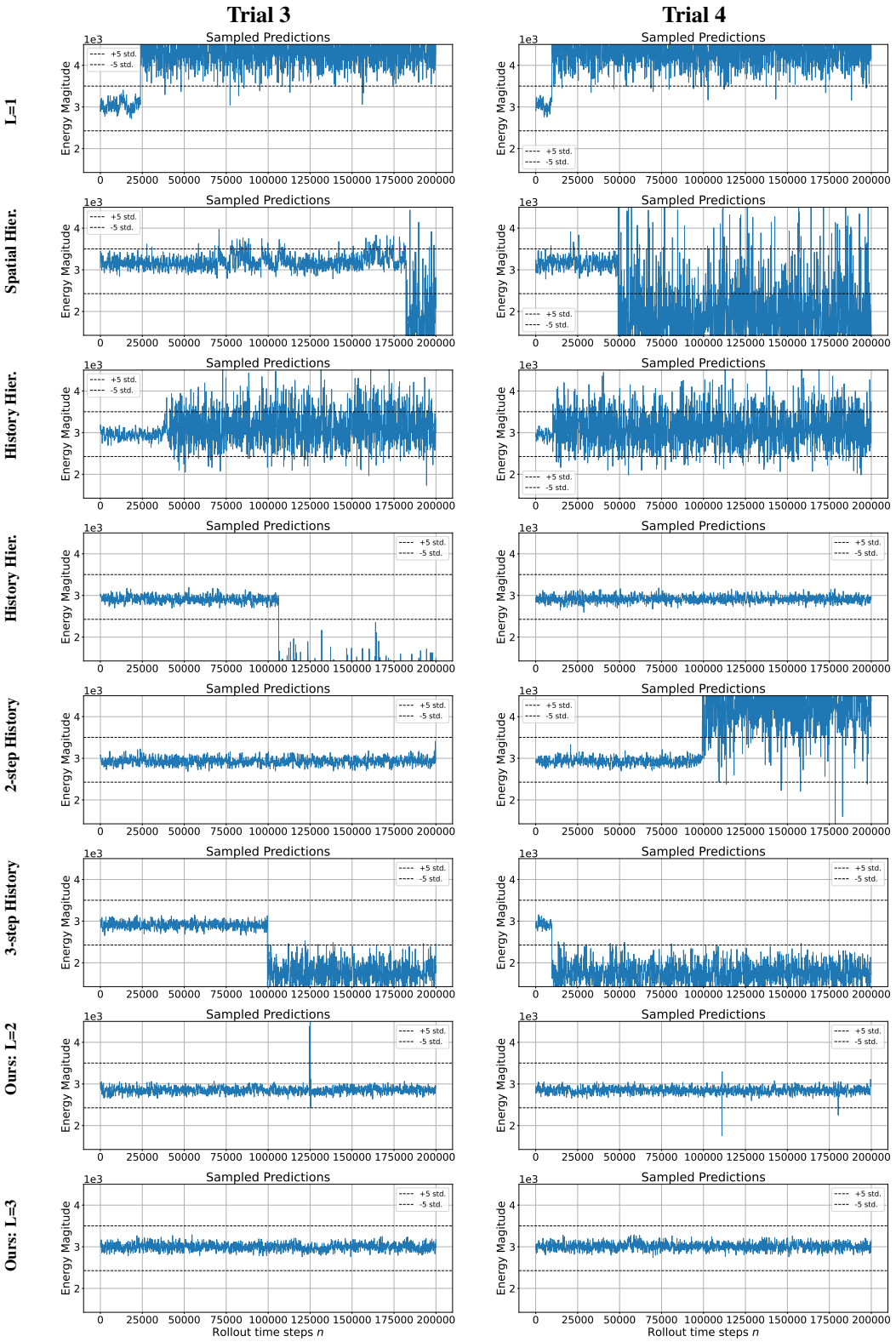


Figure 15: **Energy v.s. rollout time steps of the predicted dynamics.** The dashed horizontal line shows the maximum and minimum values computed using 5 standard deviations from the mean of the true dynamics. Our $L = 3$ method is able to maintain energy along the long-term predictions.

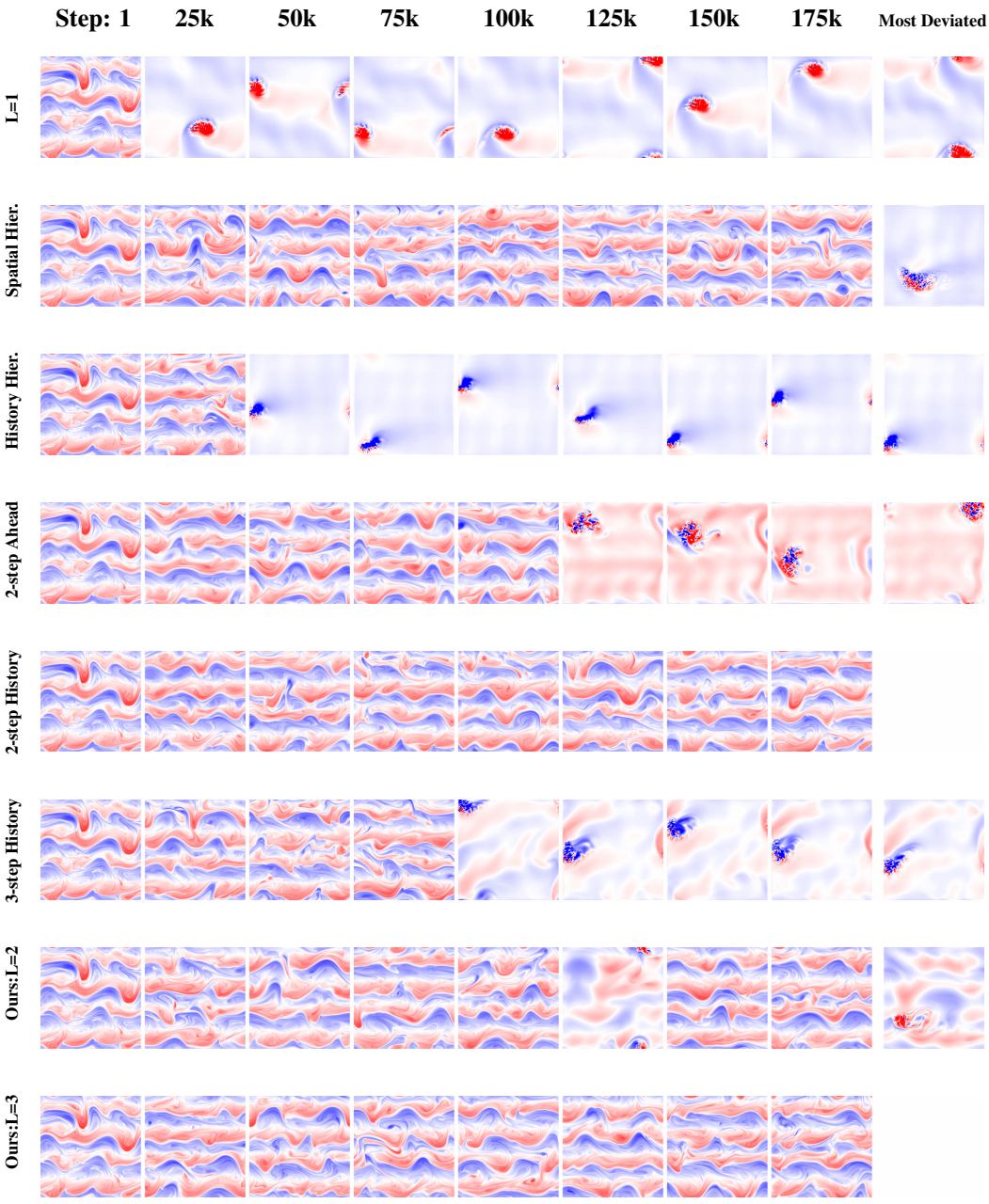


Figure 16: **Visualization of long-term rollout.** We visualize the long-term dynamics corresponds to trial 3 in Fig. 15. The rightmost column shows the frame with the maximum deviation—exceeding ± 5 standard deviations from the reference truth statistics, with a blank block indicating that all states along the sampled trajectory remain within this range. The results show that when the energy is deviated from the reference truth distribution, the dynamics exhibit exploding or averaging non-physical dynamics. Among all comparison methods, our $L = 3$ gives the most stable predictions.

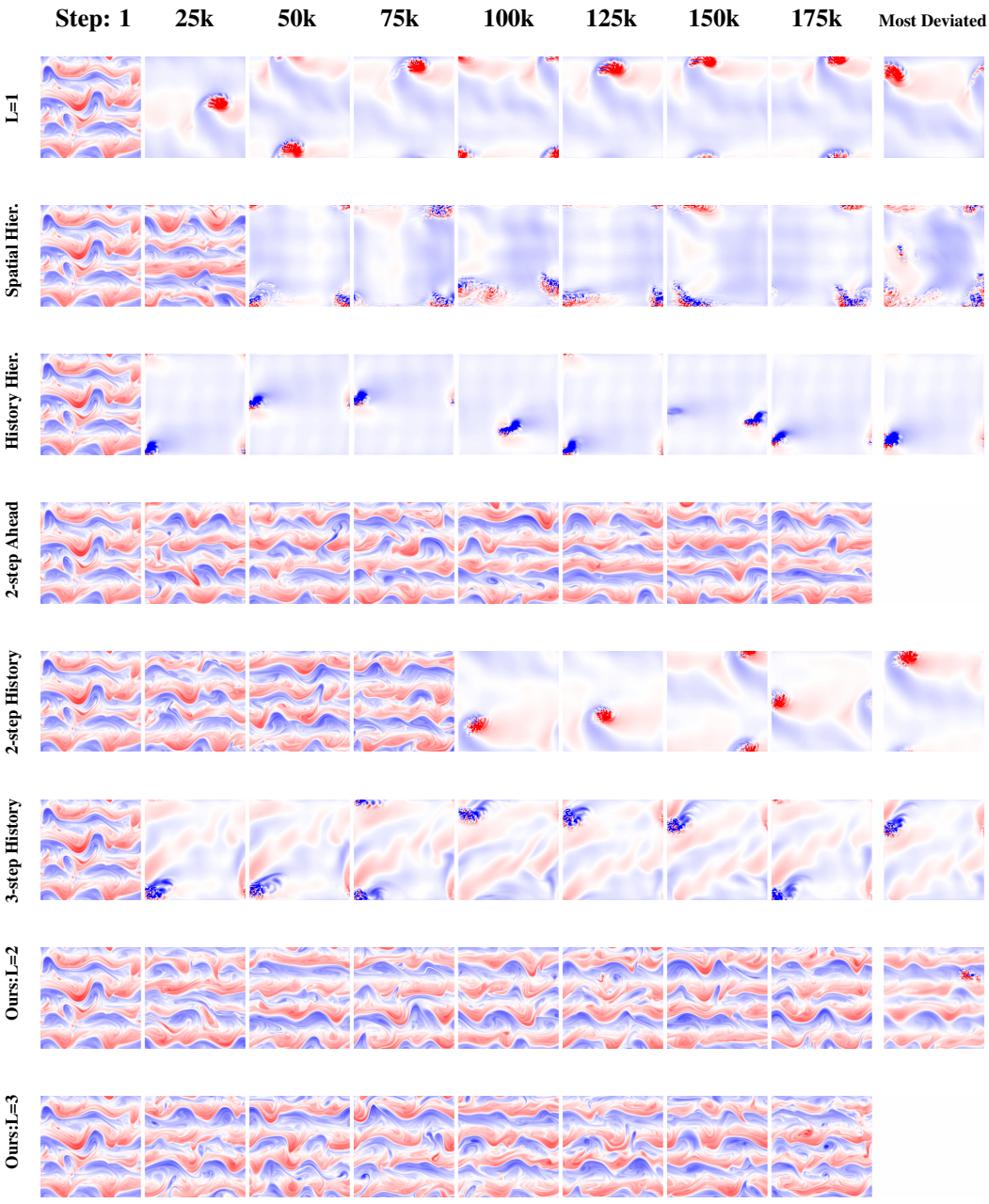


Figure 17: **Visualization of long-term rollout.** We visualize the long-term dynamics corresponds to trial 4 in Fig. 15. The rightmost column shows the frame with the maximum deviation—exceeding ± 5 standard deviations from the reference truth statistics, with a blank block indicating that all states along the sampled trajectory remain within this range. The results show that when the energy is deviated from the reference truth distribution, the dynamics exhibit exploding or averaging non-physical dynamics. Among all comparison methods, our $L = 3$ gives the most stable predictions.