SWIN-DeepONet: A Swin Transformer Enhanced DeepONet for Learning Wave Dynamics

Parth Sethi
Indian Institute of Technology Kharagpur
parth.sethi.24@kgpian.iitkgp.ac.in

Miguel Beneitez
The University of Manchester
miguel.beneitez@manchester.ac.uk

Tim Tang
The University of Manchester
tim.tang@manchester.ac.uk

Anirbit Mukherjee*
The University of Manchester
anirbit.mukherjee@manchester.ac.uk

Abstract

Accurate short-horizon prediction of nonlinear wave dynamics is critical for the safe operation of offshore infrastructure, marine vessels, and coastal systems. While neural operator frameworks such as DeepONet offer a principled approach to learning solution operators for parametric PDEs, their standard branch networks struggle to encode the coherent, multi-scale spatial structure characteristic of dispersive wave envelopes. We introduce **SWIN-DeepONet**, which replaces the MLP branch of DeepONet with a Swin Transformer encoder. By lifting the 1-D input profile into a 2-D token grid and applying hierarchical shifted-window self-attention, the Swin branch captures local-to-global spatial dependencies at linear computational cost. We evaluate both models on single-step and fully autoregressive rollout tasks using wave height envelope data governed by the Modified Nonlinear Schrödinger (MNLS) equation. SWIN-DeepONet reduces average rollout MSE by 24.3% and final relative L^2 error by 16.0% over hand-crafted Fourier-feature using DeepONet baselines, while exhibiting substantially better generalisation, with no observable train–test divergence. We also show that erstwhile strong baselines like the DINO entirely fail on this task. Thus our SWIN enhanced DeepONet opens the door to more physically faithful surrogate models for nonlinear wave dynamics.

1 Introduction

Time prediction of nonlinear dynamical systems is critical for many-query applications and real-time decision making. This task becomes particularly challenging if the dynamical systems under study are chaotic, where infinitesimal differences in the initial conditions lead to entirely different solutions after a sufficiently long time. This property of chaos establishes an horizon for predictability given by the *Lyapunov time*. The difficulty to predict chaotic systems imposes a two-fold need when developing surrogate models of such systems: (i) point-wise accurate short-time predictions and (ii) statistically-accurate long-time predictions. In this work we are motivated by oceanographic applications, where short-horizon predictions are particularly critical.

Short-horizon prediction of nonlinear sea states underpins the safe operation of offshore renewable energy assets, marine vessels and coastal infrastructure (Rusu and Onea, 2018; Faltinsen, 1993). Of particular concern is the prediction of *extreme* (rogue) waves, whose crests far exceed standard linear design envelopes and impose loads well beyond those typically considered in structural codes (Dysthe et al., 2008; Kharif et al., 2009; Onorato et al., 2013). Their formation is driven by nonlinear four-wave resonance and the Benjamin–Feir modulational instability (Benjamin and Feir, 1967), canonically described by the (modified) nonlinear Schrödinger equation for narrow-banded deep-water envelopes (Dysthe, 1979; Trulsen and Dysthe, 1996; Zakharov, 1968). High-fidelity potential-flow and Navier–Stokes solvers (Engsig-Karup et al., 2009; Jasak et al., 2007; Crespo et al., 2015) resolve this physics but are too costly for real-time forecasting or large design sweeps, motivating fast data-driven surrogates.

Machine-learning approaches to wave prediction have grown rapidly. Early work applied SVMs and feed-forward/recurrent networks to point-wise significant wave height forecasting from buoy and reanalysis data (Mafi and Amirinia, 2017; James et al., 2018; Vieira et al., 2020), while phase-resolved spatio-temporal forecasting has been tackled with CNN/LSTM hybrids (Liu et al., 2022; Klushin et al., 2021). Reduced-order and ML-assisted detectors have also been proposed specifically for rogue-wave precursor identification

*corresponding author

(Cousins and Sapsis, 2016; Sapsis, 2021).

Neural operators learn mappings between infinite-dimensional function spaces, enabling mesh-independent surrogate models for parametric PDEs (Li et al., 2021; Lu et al., 2021; Jin et al., 2022). DeepONet (Lu et al., 2021) factorizes this mapping into a *branch* network that encodes the input function at a set of sensor locations and a *trunk* network that encodes the query coordinates, combining them via an inner product. While effective, the MLP branch treats each sensor value independently and relies entirely on depth to model spatial structure — which in light of the presented experiments here can be a bottleneck when the input profile exhibits coherent oscillatory or dispersive patterns spanning many spatial scales.

Simultaneously, Vision Transformers have deeply impacted representation learning for structured 2-D signals, and the Swin Transformer (Liu et al., 2021) in particular achieves strong local-to-global feature aggregation through a hierarchical shifted-window attention mechanism at linear computational cost. Both properties suit the encoding of spatially structured PDE inputs.

This paper introduces **SWIN-DeepONet**, which substitutes the branch MLP of the standard DeepONet with a Swin Transformer encoder to account for the multi-scale spatial features of nonlinear waves. This approach is shown to supersede various alternative ideas when deployed to time-evolve wave height data.

Organization Section 2 reviews the two building blocks of our architecture: the Deep Operator Network framework (§2.1) and the Swin Transformer (§2.2) and an existing strong baseline model of DINO. Section 3 describes our Fourier-Feature DeepONet baseline in details which uses handcrafted features and serves as a stepping stone towards the SWIN-DeepONet. Section 4 introduces SWIN-DeepONet detailing the 1-D-to-2-D branch embedding (§4.1) and the paired W-MSA/SW-MSA Swin blocks (§4.2). Section 5 describes the MNLS wave-envelope dataset and checkerboard partition scheme (§5.1) for defining the test-train split, and reports hyperparameter configurations for both models (§5.2). Section 6 presents the full comparative evaluation, covering training and test loss convergence, autoregressive rollout error, spatiotemporal heatmaps, and frequency-domain spectral diagnostics. Appendix A presents the DINO architecture and its catastrophic autoregressive failure on this wave height prediction task.

2 Background

2.1 Deep Operator Networks

DeepONet (Lu et al., 2021) approximates a nonlinear operator $\mathcal{G} : \mathcal{U} \to \mathcal{V}$ by

$$\hat{\mathcal{G}}_\theta(u)(\boldsymbol{x}) = \sum_{k=1}^{q} b_k(u)\, T_k(\boldsymbol{x}), \tag{1}$$

where the *branch* $\{b_k\}$ depends only on discrete evaluations of u and the *trunk* $\{T_k\}$ depends only on the output coordinate $\boldsymbol{x}$. In practice both networks are MLPs trained end-to-end via mean-squared error.

2.2 Swin Transformer

The Swin Transformer (Liu et al., 2021) partitions a 2-D feature map into non-overlapping $M \times M$ windows and restricts self-attention to tokens within each window, reducing complexity from $\mathcal{O}((HW)^2)$ to $\mathcal{O}(HW)$. Alternating *regular* (W-MSA) and *shifted* (SW-MSA) window partitions allow information to flow across window boundaries without explicit global attention. Relative position biases are added to attention logits to encode spatial relationships among tokens within a window.

2.3 DINO

DINO (Wu et al., 2025) is a neural operator designed for long-term rollout of spatiotemporal PDE solutions. It maps a discretised spatial field $\tilde{\boldsymbol{\rho}}(t_i) \in \mathbb{R}^{N_x}$ to the next field $\tilde{\boldsymbol{\rho}}(t_{i+1})$ via a residual update that composes four operators: a pointwise lifting layer, an 8-layer full-attention Transformer acting as a global integral operator, four zero-sum-constrained convolutional blocks acting as a local differential operator, and a pointwise projection MLP. We evaluate DINO on the same wave-prediction task and find that, despite strong one-step training convergence, it suffers catastrophic autoregressive instability. Full architectural details and a failure analysis are given in Appendix A.

3 Fourier-Feature DeepONet

3.1 Overall Architecture

The Fourier-Feature DeepONet baseline follows the standard DeepONet factorisation (Section 2.1) but augments both branch and trunk networks with

a fixed Fourier feature embedding before the MLP layers. The operator prediction retains the inner-product form:

$$\hat{\mathcal{G}}_\theta\big(\tilde{\boldsymbol{\rho}}(t_i)\big)(\boldsymbol{x}_j) = \sum_{k=1}^{q} \mathcal{B}_k\big(\tilde{\boldsymbol{\rho}}(t_i)\big) \cdot \mathcal{T}_k(t_{i+1}, \boldsymbol{x}_j), \quad (2)$$

where $\theta = \{\theta_{\mathcal{B}}, \theta_{\mathcal{T}}\}$ denotes all learnable parameters.

3.2 Fourier Feature Embedding

For an input $\mathbf{z} \in \mathbb{R}^d$, the shared embedding $\phi : \mathbb{R}^d \to \mathbb{R}^{2m}$ is:

$$\phi(\mathbf{z}) = \big[\sin(2\pi \mathbf{W}_1 \mathbf{z}),\ \cos(2\pi \mathbf{W}_2 \mathbf{z})\big], \quad (3)$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{m \times d}$ are *fixed* at initialisation with entries $[\mathbf{W}]_{ij} \sim \mathcal{N}(0, \sigma^2 \omega_0^2)$ and are not updated during training. The hyperparameters (ω_0, σ) and the number of frequency pairs m are set separately for the branch and trunk, as described below.

3.3 Branch Network

The branch $\mathcal{B} : \mathbb{R}^{N_x} \to \mathbb{R}^q$ takes the full normalised spatial profile $\tilde{\boldsymbol{\rho}}(t_i) \in \mathbb{R}^{N_x}$ as input ($d = N_x$). With $m = 64$ frequency pairs and hyperparameters $(\omega_0, \sigma) = (0.5,\ 1.0)$, the embedding ϕ yields a 128-dimensional feature vector. Two hidden layers of width 256 with Gaussian Cosine Unit (GCU) activations $\sigma_{\mathrm{GCU}}(x) = x\cos(x)$ then produce $q = 50$ basis coefficients:

$$\mathbf{b}_i = \mathcal{B}\big(\tilde{\boldsymbol{\rho}}(t_i)\big) \in \mathbb{R}^q. \quad (4)$$

3.4 Trunk Network

The trunk $\mathcal{T} : \mathbb{R}^2 \to \mathbb{R}^q$ takes the query coordinates $(t_{i+1}, \boldsymbol{x}_j) \in \mathbb{R}^2$ as input ($d = 2$). Higher-frequency spatial variation is captured by setting $(\omega_0, \sigma) = (5.0,\ 10.0)$, with $m = 32$ pairs and MLP architecture as the branch, outputting q basis functions:

$$\mathbf{T}_{ij} = \mathcal{T}(t_{i+1}, \boldsymbol{x}_j) \in \mathbb{R}^q. \quad (5)$$

3.5 Training Objective

The model minimises the mean-squared error over all training pairs and spatial query locations:

$$\mathcal{L}(\theta) = \frac{1}{N_{\mathrm{train}}\, N_x} \sum_{i=1}^{N_{\mathrm{train}}} \sum_{j=1}^{N_x} \Big(\mathbf{b}_i^{\top} \mathbf{T}_{ij} - \tilde{\rho}(t_{i+1})(\boldsymbol{x}_j)\Big)^2. \quad (6)$$

All spatial outputs and input profiles are normalised to zero mean and unit variance before training and denormalised for evaluation. SWIN-DeepONet (Section 4) retains this trunk and training objective unchanged, replacing only the branch network.

4 SWIN-DeepONet

SWIN-DeepONet replaces the MLP branch of the standard DeepONet with a Swin Transformer encoder, retaining the trunk network and the inner-product output form. Given a discretised input profile $\tilde{\boldsymbol{\rho}}(t_i) \in \mathbb{R}^{N_x}$ and an output query $(t_{i+1}, \boldsymbol{x}_j)$, the prediction is:

$$\hat{\mathcal{G}}_\theta(\tilde{\boldsymbol{\rho}}(t_i))\,(\boldsymbol{x}_j) = \sum_{k=1}^{q} b_{i,k}\, T_{ij,k}, \quad (7)$$

where the branch coefficients $\mathbf{b}_i \in \mathbb{R}^q$ are produced by a Swin encoder and $\mathbf{T}_{ij} \in \mathbb{R}^q$ by a Fourier-feature MLP trunk. The two sub-networks are described in turn below.

4.1 Branch: 1-D to 2-D Embedding

The 1-D sensor array $\tilde{\boldsymbol{\rho}}(t_i) \in \mathbb{R}^{N_x}$ must be cast into a 2-D spatial representation before it can be processed by the Swin blocks. We zero-pad and reshape the signal into a single-channel grid of size $H_{\mathrm{p}} \times W_{\mathrm{p}}$, where

$$H_{\mathrm{p}} = W_{\mathrm{p}} = M\left\lceil \frac{\lceil \sqrt{N_x} \rceil}{M} \right\rceil, \qquad M = 7, \quad (8)$$

ensuring exact divisibility by the window size M. Each scalar cell is then linearly projected to an embedding dimension $C = 64$ via a 1×1 convolution, followed by Layer Normalisation, yielding the initial token sequence $\mathbf{Z}^{(0)} \in \mathbb{R}^{H_{\mathrm{p}} W_{\mathrm{p}} \times C}$.

This embedding strategy is architecture-agnostic — any fixed-length 1-D physics signal can be lifted to a compatible 2-D representation by padding to the nearest integer multiple of M^2 and reshaping.

4.2 Swin Transformer Blocks

Two successive Swin blocks process $\mathbf{Z}^{(0)}$. Block 1 applies regular window-based multi-head self-attention (W-MSA, shift $s = 0$); Block 2 applies shifted-window MSA (SW-MSA, shift $s = \lfloor M/2 \rfloor = 3$). Both blocks share the same hyperparameters: $n_h = 4$ attention heads, head dimension $d = C/n_h = 16$, and an intra-block MLP of hidden width $4C = 256$ with GELU activations. With

pre-normalisation, the paired update equations are:

$$\begin{aligned}\hat{\mathbf{Z}}^{(l)} &= \text{W-MSA}\big(\text{LN}(\mathbf{Z}^{(l-1)})\big) + \mathbf{Z}^{(l-1)},\\ \mathbf{Z}^{(l)} &= \text{MLP}\big(\text{LN}(\hat{\mathbf{Z}}^{(l)})\big) + \hat{\mathbf{Z}}^{(l)},\\ \hat{\mathbf{Z}}^{(l+1)} &= \text{SW-MSA}\big(\text{LN}(\mathbf{Z}^{(l)})\big) + \mathbf{Z}^{(l)},\\ \mathbf{Z}^{(l+1)} &= \text{MLP}\big(\text{LN}(\hat{\mathbf{Z}}^{(l+1)})\big) + \hat{\mathbf{Z}}^{(l+1)}.\end{aligned} \tag{9}$$

4.3 Window-Based Attention

The feature map is partitioned into $n_W = (H_\text{p}/M)^2$ non-overlapping $M \times M$ windows of $M^2 = 49$ tokens each. For a window $\mathbf{w} \in \mathbb{R}^{M^2 \times C}$, each head h obtains its queries, keys, and values via a dedicated projection $\mathbf{W}_{qkv}^{(h)} \in \mathbb{R}^{C \times 3d}$, giving $\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}^{(h)} \in \mathbb{R}^{M^2 \times d}$. The attention logit matrix for head h is:

$$\mathbf{A}^{(h)} = \frac{\mathbf{Q}^{(h)}\big(\mathbf{K}^{(h)}\big)^\top}{\sqrt{d}} + \mathbf{B}^{(h)}, \quad \mathbf{A}^{(h)} \in \mathbb{R}^{M^2 \times M^2}, \tag{10}$$

where $\mathbf{B}^{(h)}$ is the *relative position bias*. For each token pair (i, j) within a window, relative height and width offsets are $\Delta_h, \Delta_w \in [-(M-1), M-1]$. These are shifted to non-negative indices $\tilde{\Delta}_h, \tilde{\Delta}_w \in [0, 2M-2]$ and encoded as a scalar:

$$\text{idx}(i,j) = \tilde{\Delta}_h(i,j) \cdot (2M-1) + \tilde{\Delta}_w(i,j). \tag{11}$$

A learnable table $\hat{\mathbf{B}} \in \mathbb{R}^{(2M-1)^2 \times n_h} = \mathbb{R}^{169 \times 4}$ gives $B_{ij}^{(h)} = \hat{B}_{\text{idx}(i,j),h}$. The multi-head output is:

$$\begin{aligned}\text{Attn}^{(h)} &= \text{Softmax}\big(\mathbf{A}^{(h)}\big)\mathbf{V}^{(h)},\\ \text{W-MSA}(\mathbf{w}) &= \big[\text{Attn}^{(1)}; \cdots; \text{Attn}^{(n_h)}\big]\mathbf{W}_o.\end{aligned} \tag{12}$$

The windowed scheme achieves *linear* complexity $\Omega(\text{W-MSA}) = 4H_\text{p}W_\text{p}C^2 + 2M^2H_\text{p}W_\text{p}C$, versus the quadratic $\Omega(\text{MSA}) = 4H_\text{p}W_\text{p}C^2 + 2(H_\text{p}W_\text{p})^2C$ of global attention.

4.4 Shifted Window and Attention Mask

SW-MSA cyclically shifts the feature map by $(-s, -s)$ before partitioning:

$$\tilde{\mathbf{Z}}_{h,w} = \mathbf{Z}_{(h-s) \bmod H_\text{p},\,(w-s) \bmod W_\text{p}}. \tag{13}$$

This brings tokens from adjacent windows into the same local window, enabling cross-window information flow without additional attention operations. To prevent spurious attention between spatially non-adjacent tokens that are co-located after the shift, an additive mask is applied to the logits before softmax:

$$m_{ij} = \begin{cases} 0 & i, j \text{ in same sub-window},\\ -100 & \text{otherwise}, \end{cases} \tag{14}$$

The mask is constructed by labelling the $3 \times 3 = 9$ spatial regions induced by the three slice boundaries in each dimension (arising from the cyclic shift). The inverse shift is applied after window merging to restore the original spatial layout.

4.5 Pooling and Branch Projection

After Layer Normalisation, global average pooling collapses the spatial dimension of the final token sequence $\mathbf{Z}^{(2)}$:

$$\bar{\mathbf{z}} = \frac{1}{H_\text{p}W_\text{p}} \sum_{h,w} \mathbf{Z}_{h,w}^{(2)} \in \mathbb{R}^C. \tag{15}$$

A linear projection then maps the pooled representation to the $q = 50$ basis coefficients required by Equation 7:

$$\mathbf{b}_i = \mathbf{W}_\text{proj}\,\bar{\mathbf{z}} + \mathbf{b}_\text{proj}, \qquad \mathbf{W}_\text{proj} \in \mathbb{R}^{q \times C}. \tag{16}$$

4.6 Trunk Network

The trunk maps a query $(t_{i+1}, \boldsymbol{x}_j) \in \mathbb{R}^2$ to $\mathbb{R}^q$ via a Fourier feature embedding followed by a two-hidden-layer MLP. With $q_\text{ff} = 32$ frequency pairs, the embedding is:

$$\phi(\mathbf{z}) = \big[\sin(2\pi\mathbf{W}_1\mathbf{z}),\ \cos(2\pi\mathbf{W}_2\mathbf{z})\big] \in \mathbb{R}^{64}, \tag{17}$$

where $[\mathbf{W}_k]_{ij} \sim \mathcal{N}(0, \sigma^2\omega_0^2)$, with $\omega_0 = 5.0$ and $\sigma = 10.0$. The frequency matrices $\mathbf{W}_1, \mathbf{W}_2$ are fixed at initialisation and not updated during training. The MLP uses Gaussian-Cosine Unit (GCU) activations, $\sigma_\text{GCU}(x) = x\cos(x)$, and two hidden layers of width 256. Setting $\mathbf{h}^{(0)} = \phi(t_{i+1}, \boldsymbol{x}_j)$, the forward pass is:

$$\mathbf{h}^{(k)} = \sigma_\text{GCU}\big(\mathbf{W}_T^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}_T^{(k)}\big), \quad k = 1, 2 \tag{18}$$

$$\mathbf{T}_{ij} = \mathbf{W}_T^{(3)}\mathbf{h}^{(2)} + \mathbf{b}_T^{(3)} \in \mathbb{R}^q. \tag{19}$$

4.7 Training Objective

The model minimises the mean-squared error over all training pairs and spatial query locations:

$$\mathcal{L}(\theta) = \frac{1}{N_\text{train}\,N_x} \sum_{i=1}^{N_\text{train}} \sum_{j=1}^{N_x} \Big(\mathbf{b}_i^\top \mathbf{T}_{ij} - \tilde{\rho}(t_{i+1})(\boldsymbol{x}_j)\Big)^2 \tag{20}$$

All spatial outputs and input profiles are normalised to zero mean and unit variance before training and denormalised for evaluation.

Discussions

— On Embedding design. The 1-D-to-2-D reshape is the key to model the time-series signal for a vision transformer. The window-size alignment constraint, padding to the nearest multiple of M introduces at most $M^2 - 1 = 48$ padding tokens for $M = 7$, a negligible overhead for typical PDE grids where $N_x \gg 50$.

— On Complexity. With $H_{\mathrm{p}} W_{\mathrm{p}} = \mathcal{O}(N_x)$ tokens, window attention costs $\mathcal{O}(N_x M^2 C + N_x C^2)$, which is linear in N_x for fixed M and C. This scales well as the spatial resolution increases.

5 Experimental Setup

This section describes the dataset, data partitioning scheme, and hyperparameter configurations used to train and evaluate both Fourier-Feature DeepONet and SWIN-DeepONet. All experiments share the same dataset and checkerboard partition; model-specific hyperparameters are reported separately for each architecture.

5.1 Dataset of MNLS Wave Envelopes

We evaluate on a dataset of complex wave-envelope profiles governed by the modified nonlinear Schrödinger (MNLS) equation, which models weakly nonlinear deep-water surface gravity waves. Each trajectory consists of a time series of spatial snapshots $\{\tilde{\rho}(t_i)\}_{i=0}^{T}$, where $\tilde{\rho} \in \mathbb{R}^{N_x}$ is the discretised envelope amplitude on a uniform spatial grid of N_x points. The task is single-step prediction: given the profile at time t_i, predict the profile at t_{i+1}.

The dataset is split into disjoint training and test trajectories using a checkerboard partition that ensures no time-step pair from a test trajectory leaks into the training set.

5.2 Hyperparameters

Hyperparameters for both models are summarised in Tables 1 and 2. The trunk network is identical across both architectures; only the branch differs.

6 Results

We compare Swin-DeepONet and Fourier-Feature DeepONet against DINO on the held-out checkerboard test split $\mathcal{X}_{\mathrm{test}}$. All three models are evaluated under *autoregressive rollout*: the rollout is initialised from the true first test-time profile $\hat{\rho}^{(0)} = \tilde{\rho}(t_0^{\mathrm{test}})$ and advanced recursively as $\hat{\rho}^{(k+1)} = \hat{\mathcal{G}}_\theta(\hat{\rho}^{(k)})$, making this a strictly harder evaluation than teacher-forced testing.

Table 1: Fourier-Feature DeepONet hyperparameters.

Component	Parameter	Value
Branch (MLP)	FF frequencies m	64
	(ω_0, σ)	$(0.5,\ 1.0)$
	Hidden width / layers	256 / 2
	Activation	$x\cos(x)$
	Output dim q	50
Trunk (MLP)	FF frequencies q_{ff}	32
	(ω_0, σ)	$(5.0,\ 10.0)$
	Hidden width / layers	256 / 2
	Activation	$x\cos(x)$
	Output dim q	50

Table 2: SWIN-DeepONet hyperparameters.

Component	Parameter	Value
Branch (Swin)	Patch size	1×1
	Embedding dim C	64
	Swin blocks L	2
	Attention heads n_h	4
	Head dim $d = C/n_h$	16
	Scale $\delta = d^{-1/2}$	0.25
	Window size M	7
	Shift size s	3
	Rel. pos. table $\hat{\mathbf{B}}$	$\mathbb{R}^{169\times 4}$
	MLP hidden / expansion	256 / 4×
	MLP activation	GELU
	Output dim q	50
Trunk (MLP)	FF frequencies q_{ff}	32
	(ω_0, σ)	$(5.0,\ 10.0)$
	Hidden width / layers	256 / 2
	Activation	$x\cos(x)$
	Output dim q	50

The aggregate rollout metrics are summarised in Table 3. DINO diverges catastrophically under autoregressive rollout despite strong one-step training convergence (see Appendix A), reaching a relative L^2 error exceeding 6000% by step 1316. Among the stable models, Swin-DeepONet reduces average rollout MSE by 24.3% and final relative L^2 error by 16.0% over the Fourier-Feature baseline.

Figure 1 shows the training loss (Eqs. 6, 20, 43), evaluated at every epoch on both the training and held-out test partitions. The three models exhibit qualitatively different generalisation behaviour. DINO converges most rapidly, reaching ~ 3–5×10^{-5} with no observable train–test gap, yet this apparent one-step success masks catastrophic autoregressive instability (Appendix A).

Table 3: Summary metrics for all models. Relative L^2 rollout error (%) is reported at increasing autoregressive step counts. Avg. rollout MSE and final relative L^2 are computed over the full 1316-step rollout for the DeepONet variants. All models use the checkerboard partition. Lower is better.

Model	**1-step**	**60-step**	**99-step**	**1316-step**
	Relative L^2 error (%), lower is better			
DINO	7.27	2967.96	4334.30	6164.51
FF-DeepONet	80.41	76.01	76.72	76.75
Swin-DeepONet	**71.54**	**63.65**	**64.37**	**67.12**

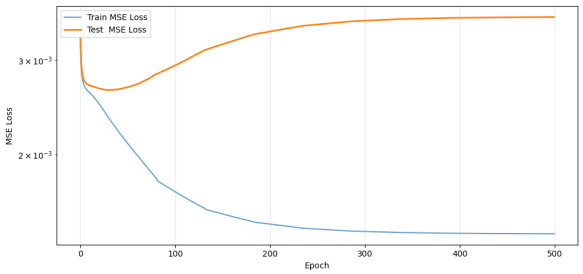
Model	**Avg. rollout MSE**	**Final rel. L^2**
DINO	19.38	6431.33
FF-DeepONet	2.77×10^{-3}	0.726
Swin-DeepONet	$\mathbf{2.10\times10^{-3}}$	**0.610**

For Fourier-Feature DeepONet, the training loss decreases monotonically throughout all 500 epochs, ultimately falling below 1×10^{-3}, while the test loss diverges after an initial co-descent near epoch 25 and plateaus at approximately 3.5×10^{-3}; the resulting train–test gap is evidence of overfitting to the training spatial and temporal coordinates. Swin-DeepONet displays a qualitatively different regime: after a sharp transient in the first epoch, train and test losses remain nearly indistinguishable throughout training, both declining jointly and smoothly from 2.7×10^{-3} to 2.2×10^{-3} by epoch 500 with no visible divergence. *This indicates that the Swin branch generalises substantially better to unseen checkerboard coordinates, consistent with its lower aggregate rollout error in Table 3.*

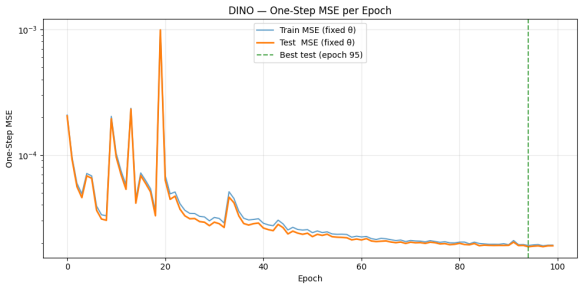
Autoregressive Rollout Error (Figure 2). Figure 2 plots the relative L^2 rollout error

$$\varepsilon_{L^2}^{(k)} = \frac{\|\hat{\rho}^{(k)} - \tilde{\rho}(t_k)\|_2}{\|\tilde{\rho}(t_k)\|_2}, \tag{21}$$

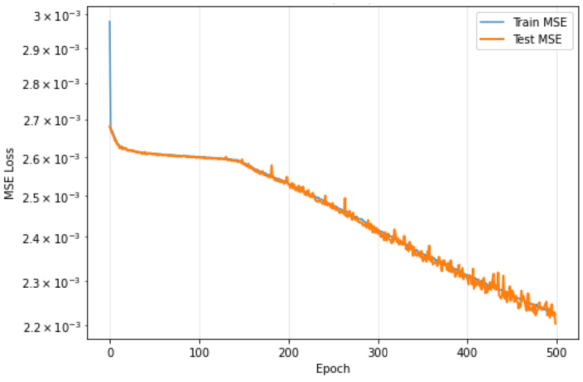
at each autoregressive step k, where $\hat{\rho}^{(k+1)} = \hat{\mathcal{G}}_\theta(\hat{\rho}^{(k)})$ is advanced from the true initial test profile $\hat{\rho}^{(0)} = \tilde{\rho}(t_0^{\text{test}})$ with no ground-truth injection at any subsequent step. Normalising by $\|\tilde{\rho}(t_k)\|_2$ makes the metric insensitive to the overall amplitude scale of the wave envelope, so differences between curves reflect genuine structural prediction error rather than amplitude drift. DINO diverges exponentially and exits the plot range within the first 50 steps; its full failure trajectory is shown in Appendix A. Both DeepONet variants undergo a sharp transient in the first ~ 50 steps before settling into a stationary oscillatory regime with no subsequent monotonic growth, confirming that the learned dynamics are bounded.



(a) Fourier-Feature DeepONet



(b) DINO



(c) Swin-DeepONet

Figure 1: Training and test loss (Eqs. 6, 20, 43) versus epoch for both DeepONet variants and DINO. Blue lines denote training loss; Orange lines denote test loss.

After this transient, Swin-DeepONet (orange) occupies a consistently lower error band of approximately 0.62–0.75 across all 3200 rollout steps, whereas Fourier-Feature DeepONet (blue) oscillates in a higher band of roughly 0.72–0.90, with more pronounced upward spikes. *The consistent separation across 3200 steps confirms that the Swin branch yields a uniformly more accurate long-horizon predictor than both Fourier-Feature DeepONet and DINO.*

Spatiotemporal Prediction Heatmaps (Figure 3). Figure 3 displays the full space–time prediction quality for the two DeepONet variants; DINO's

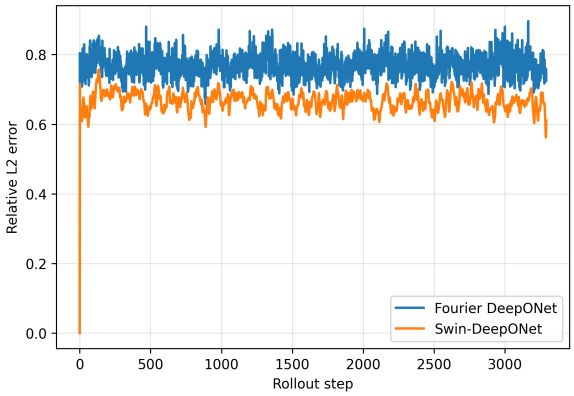


Figure 2: Relative L^2 rollout error $\varepsilon_{L^2}^{(k)}$ (Equation 21) versus autoregressive step k for Fourier-Feature DeepONet (blue) and Swin-DeepONet (orange). DINO diverges beyond the plot range within the first 50 steps and is omitted for scale. Both DeepONet models are initialised from the true test profile at $k = 0$ with no subsequent teacher forcing.

heatmaps are shown in Appendix A as they reflect a degenerate prediction. Each row shows, from left to right: the ground-truth intensity field $\tilde{\rho}(t_k, x_j)$, the autoregressive prediction $\hat{\rho}^{(k)}(x_j)$, and the signed pointwise error

$$e_j^{(k)} = \hat{\rho}^{(k)}(x_j) - \tilde{\rho}(t_k, x_j), \tag{22}$$

plotted as a function of spatial coordinate x_j (vertical axis) and rollout step k (horizontal axis). The ground truth is characterised by coherent diagonal striations spanning the full domain, punctuated by sparse high-amplitude events reaching $\approx 6\times10^{-2}$ corresponding to propagating near-rogue wave packets. Fourier-Feature DeepONet produces a spatially diffuse prediction that almost entirely loses this diagonal structure, indicating that the MLP branch cannot encode the coherent propagating envelope. Swin-DeepONet visibly recovers the diagonal banding, with striations consistent in orientation and spacing with the ground truth. Both models underpredict the peak amplitudes of extreme events. The error panels, on a symmetric logarithmic scale, confirm this picture: the Fourier residual is dominated by large coherent striations mirroring the missed propagating structure, whereas the Swin residual shows the same pattern at reduced amplitude, consistent with its partial recovery of the envelope.

The Fourier-Feature DeepONet residual reaches a peak absolute error of 6.29×10^{-2}, with a mean absolute error of 2.43×10^{-3} and an RMS error

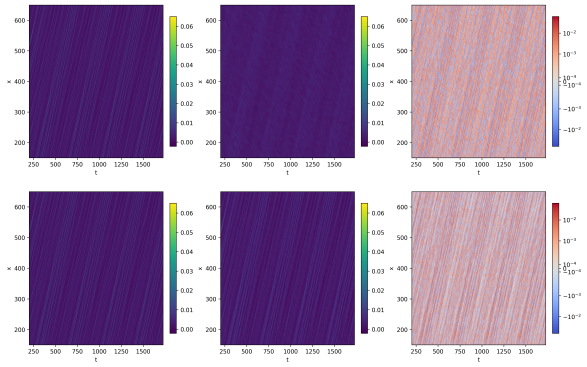


Figure 3: Space–time heatmaps of (left) ground-truth intensity $\tilde{\rho}(t_k, x_j)$, (centre) autoregressive prediction $\hat{\rho}^{(k)}(x_j)$, and (right) signed pointwise error $e_j^{(k)}$ (Equation 22), for Fourier-Feature DeepONet (top) and Swin-DeepONet (bottom). The error panels share a symmetric logarithmic scale.

of 3.50×10^{-3}. Swin-DeepONet reduces all three: peak error 5.63×10^{-2} (10.5% lower), mean absolute error 2.09×10^{-3} (13.9% lower), and RMS error 3.05×10^{-3} (13.0% lower).

Frequency-Domain Diagnostics (Figure 4). Figure 4 presents two spectral diagnostics derived from the two-dimensional discrete Fourier transforms $\hat{F}[k_t, k_x]$ and $F[k_t, k_x]$ of the predicted and ground-truth space–time fields, respectively. Let $\mathcal{M}_\delta = \{\mathbf{k} : 0 < \|\mathbf{k}\| \le \delta\}$ denote the ball of frequency indices within radius δ of the origin (excluding the zero-frequency component), where $\|\mathbf{k}\| = \sqrt{\nu_t^2 + \nu_x^2}$ is the Euclidean norm of the normalised frequency coordinates $(\nu_t, \nu_x) \in [-\frac{1}{2}, \frac{1}{2})^2$, and let $\mathcal{M}_\delta^* \subseteq \mathcal{M}_\delta$ be the further restriction to modes for which $F[\mathbf{k}] \neq 0$. The *relative nonzero-mode Fourier gap*:

$$\varepsilon_{\text{rel}}^{(\delta)} = \frac{\sum_{\mathbf{k}\in\mathcal{M}_\delta^*} |\hat{F}[\mathbf{k}] - F[\mathbf{k}]|}{\sum_{\mathbf{k}\in\mathcal{M}_\delta^*} |F[\mathbf{k}]|}, \tag{23}$$

measures the cumulative fractional amplitude discrepancy over all energetically active modes up to radius δ. The *per-mode Fourier gap*:

$$\varepsilon_{\text{mode}}^{(\delta)} = \frac{\|\hat{F} - F\|_{\ell^2(\mathcal{M}_\delta)}}{\sqrt{|\mathcal{M}_\delta|}}, \tag{24}$$

is the RMS per-coefficient discrepancy within $\mathcal{M}_\delta$, normalising out the growth in mode count as δ increases. While $\varepsilon_{\text{rel}}^{(\delta)}$ reports relative fidelity, $\varepsilon_{\text{mode}}^{(\delta)}$ reports the average absolute spectral error per

mode, making it sensitive to whether the prediction places spectral energy at the right Fourier coefficients regardless of the true field's amplitude at those scales. These diagnostics are computed for the two DeepONet variants; DINO's diverged predictions yield uninformative spectral content and are excluded.

Both panels tell a consistent story. In the left panel, both curves rise sharply to ≈ 1 by $\delta \approx 0.25$ and plateau thereafter. A value of $\varepsilon_{\text{rel}}^{(\delta)} \approx 1$ indicates that the total amplitude of the prediction error across modes in $\mathcal{M}_\delta^*$ is comparable to the total true spectral amplitude at those same modes, that is, the model's high-frequency predictions are essentially uncorrelated with the true field and carry no useful spectral information.

The critical difference lies at the lower frequencies ($\delta < 0.1$): Swin-DeepONet's relative gap is several orders of magnitude lower than Fourier-Feature DeepONet's at these scales, confirming that the Swin branch captures the dominant low-frequency propagating structure that the MLP branch misses entirely.

The per-mode gap (right panel) reinforces this picture. Fourier-Feature DeepONet reaches $\sim 10^2$ per mode at small δ, whereas Swin-DeepONet remains below 1 at the same radii; both models then converge to comparable values (~ 8) at large δ. The sharp drop at small radii for Swin-DeepONet reflects genuine recovery of the large-scale propagating modes, not merely a scale effect from averaging over more coefficients.

The spectral analysis thus corroborates the heatmap observation: Swin-DeepONet recovers the large-scale propagating envelope while both architectures equally fail to resolve the fine-scale wave structure.

Spatial Profile Snapshots (Figure 5). Figure 5 shows spatial profiles of $|U|^2$ at four rollout steps ($k = 4, 16, 64, 128$). The ground truth consists of a slowly-varying background modulation interspersed with sharp, narrow spikes whose amplitudes grow with rollout time, reaching $\approx 6 \times 10^{-2}$ by step 128. Both DeepONet variants reproduce the broad envelope shape but smooth over the sharp peaks, saturating near the background level of ~ 2–5×10^{-3}. The two predictions are nearly identical across most of the domain at all four snapshots. *At the largest peaks, however, Swin-DeepONet produces slightly higher amplitude responses that are closer to the ground truth, consistent with its better recovery of the propagating envelope structure seen in the heatmaps.*

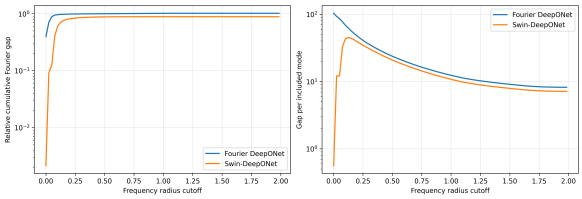


Figure 4: Frequency-domain rollout diagnostics for Fourier-Feature DeepONet and Swin-DeepONet. *Left:* Relative nonzero-mode Fourier gap $\varepsilon_{\text{rel}}^{(\delta)}$ (Equation 23) versus frequency radius δ; a value near 1 indicates prediction error comparable to true spectral amplitude at those scales. *Right:* Per-mode Fourier gap $\varepsilon_{\text{mode}}^{(\delta)}$ (Equation 24), the RMS per-coefficient discrepancy within $\mathcal{M}_\delta$. Lower curves indicate more faithful spectral reproduction.

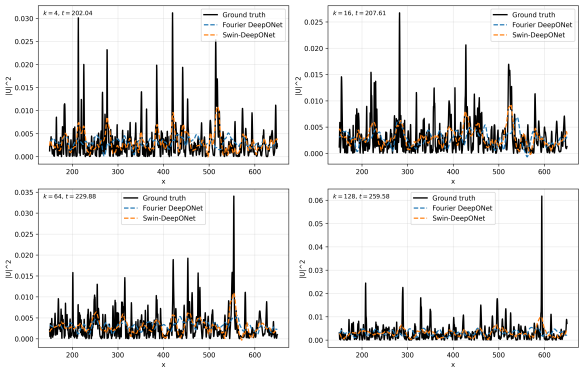


Figure 5: Instantaneous spatial profiles of $|U|^2$ at rollout steps $k = 4, 16, 64, 128$. Ground truth is shown in black; Fourier-Feature DeepONet and Swin-DeepONet are shown as blue and orange dashed lines respectively.

7 Conclusion

The results presented in this work motivate a number of next steps of research. *Firstly,* this is a first-of-its-kind step towards fast prediction of real wave data and other multiscale chaotic systems, including wall-bounded turbulence. *Secondly,* the architecture needs improvements to be able to be deployed successfully at high spatial resolution as is needed for real ocean data. Increasing resolution is particularly challenging as that steeply increases the computational cost of training the model. We posit that it would be a challenging and rewarding direction of research to find ways to efficiently scale the Swin-DeepONet architecture to match the practical requirements of being deploying in the

oceans.

Thirdly, the tests so far are restricted to solutions of the MNLS PDE, and we recognize that there are multiple models of ocean wave data. It is also recognized that ocean wave data has significant difference between different oceans and also between locations, like near coast waves vs deep ocean waves. It would be very fruitful to be able to scale experiments shown here to models which can evolve multiple such classes of waves i.e build "ocean wave foundation models".

Limitations

A clear limitation of the method so far is that the ability of our best architecture (Swin-DeepONet) is still limited at reproducing the high-frequency modes of ocean waves. Secondly, we have not yet tested the method at high spatial resolution or for waves propagating in two dimensions – which is a better model of the real oceans.

References

T. Brooke Benjamin and J. E. Feir. 1967. The disintegration of wave trains on deep water. Part 1. theory. *Journal of Fluid Mechanics*, 27(3):417–430.

Will Cousins and Themistoklis P. Sapsis. 2016. Reduced-order precursors of rare events in unidirectional nonlinear water waves. *Journal of Fluid Mechanics*, 790:368–388.

A. J. C. Crespo, J. M. Domínguez, B. D. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. García-Feal. 2015. DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH). *Computer Physics Communications*, 187:204–216.

Kristian Dysthe, Harald E. Krogstad, and Peter Müller. 2008. Oceanic rogue waves. *Annual Review of Fluid Mechanics*, 40:287–310.

Kristian B. Dysthe. 1979. Note on a modification to the nonlinear Schrödinger equation for application to deep water waves. *Proceedings of the Royal Society of London. Series A*, 369(1736):105–114.

Allan P. Engsig-Karup, Harry B. Bingham, and Ole Lindberg. 2009. An efficient flexible-order model for 3D nonlinear water waves. *Journal of Computational Physics*, 228(6):2100–2118.

Odd M. Faltinsen. 1993. *Sea Loads on Ships and Offshore Structures*. Cambridge University Press.

Scott C. James, Yushan Zhang, and Fearghal O'Donncha. 2018. A machine learning framework to forecast wave conditions. *Coastal Engineering*, 137:1–10.

Hrvoje Jasak, Aleksandar Jemcov, and Željko Tuković. 2007. OpenFOAM: A C++ library for complex physics simulations. *International Workshop on Coupled Methods in Numerical Dynamics*.

Pengzhan Jin, Shuai Meng, and Lu Lu. 2022. MIONet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514.

Christian Kharif, Efim Pelinovsky, and Alexey Slunyaev. 2009. *Rogue Waves in the Ocean*. Springer.

Andrey Klushin, Konstantin Belyaev, and Natalia Tuchkova. 2021. Deep learning for short-term ocean wave forecasting. *Journal of Marine Science and Engineering*, 9(12):1399.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2021. Fourier neural operator for parametric partial differential equations. In *ICLR*.

Yang Liu, Xiaowei Zhang, Gang Chen, and Sheng Dong. 2022. Phase-resolved wave prediction with deep learning. *Ocean Engineering*, 264:112430.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqing Zhang, and George Em Karniadakis. 2021. Learning nonlinear operators via DeepONet. *Nature Machine Intelligence*, 3:218–229.

Somayeh Mafi and Gholamreza Amirinia. 2017. Forecasting hurricane wave height in Gulf of Mexico using soft computing methods. *Ocean Engineering*, 146:352–362.

Miguel Onorato, Stefania Residori, Umberto Bortolozzo, Antonio Montina, and F. T. Arecchi. 2013. Rogue waves and their generating mechanisms in different physical contexts. *Physics Reports*, 528(2):47–89.

Eugen Rusu and Florin Onea. 2018. A review of the technologies for wave energy extraction. *Clean Energy*, 2(1):10–19.

Themistoklis P. Sapsis. 2021. Statistics of extreme events in fluid flows and waves. *Annual Review of Fluid Mechanics*, 53:85–111.

Karsten Trulsen and Kristian B. Dysthe. 1996. A modified nonlinear Schrödinger equation for broader bandwidth gravity waves on deep water. *Wave Motion*, 24(3):281–289.

Felipe Vieira, Geraldo Cavalcante, Edmo Campos, and Francisco Taveira-Pinto. 2020. Modeling significant wave heights for multiple time horizons using metaheuristic regression methods. *Ocean Engineering*, 213:107603.

Hao Wu, Yuan Gao, Fan Xu, Fan Zhang, Qingsong Wen, Kun Wang, Xiaomeng Huang, and Xian Wu. 2025.

Differential-integral neural operator for long-term turbulence forecasting. *Preprint*, arXiv:2509.21196.

V. E. Zakharov. 1968. Stability of periodic waves of finite amplitude on the surface of a deep fluid. *Journal of Applied Mechanics and Technical Physics*, 9(2):190–194.

A DINO: Architecture and Failure Analysis on Wave Data

DINO (Wu et al., 2025) directly maps a normalised discrete spatial field $\tilde{\boldsymbol{\rho}}(t_i) \in \mathbb{R}^{N_x}$ to the next normalised field $\tilde{\boldsymbol{\rho}}(t_{i+1}) \in \mathbb{R}^{N_x}$ via four sequentially composed operators acting on a latent feature space of dimension d, rather than decomposing the operator into branch and trunk networks as in DeepONet.

A.1 Data Partition

DINO uses the same checkerboard partition as the DeepONet variants. Odd-indexed timesteps and spatial points form the training set; even-indexed timesteps and spatial points form the held-out test set:

$$\mathcal{D}_{\text{train}} = \{(t_i, \mathbf{x}_j) : i \equiv 1 \pmod 2,\ j \equiv 1 \pmod 2\}, \tag{25}$$

$$\mathcal{D}_{\text{test}} = \{(t_i, \mathbf{x}_j) : i \equiv 0 \pmod 2,\ j \equiv 0 \pmod 2\}, \tag{26}$$

yielding subgrids of size $(N_t/2) \times (N_x/2)$ for each partition. Min-max normalisation is fitted exclusively on $\mathcal{D}_{\text{train}}$:

$$\tilde{\rho} = \frac{\rho - \rho_{\min}^{\text{train}}}{\rho_{\max}^{\text{train}} - \rho_{\min}^{\text{train}}}, \tag{27}$$

and applied identically to $\mathcal{D}_{\text{test}}$, preventing leakage of test statistics into the normalisation. Training pairs are consecutive one-step transitions on the subsampled time axis:

$$\mathcal{D}_{\text{pairs}} = \left\{\left(\tilde{\boldsymbol{\rho}}(t_i),\ \tilde{\boldsymbol{\rho}}(t_{i+2})\right) : i \equiv 1 \pmod 2,\ i < N_t - 2\right\}, \tag{28}$$

so the model learns to advance the field by one subsampled step $2\Delta t$, consistent with autoregressive rollout at inference time.

A.2 Architecture

The forward pass proceeds as:

$$\mathbf{z}_i^{\text{lift}} = \mathcal{L}_{\theta_L}\left(\tilde{\boldsymbol{\rho}}(t_i)\right) \in \mathbb{R}^{N_x \times d}, \tag{29}$$

$$\mathbf{z}_i^{\text{int}} = \mathcal{G}_{\theta_I}^{\text{Int}}\left(\mathbf{z}_i^{\text{lift}}\right) \in \mathbb{R}^{N_x \times d}, \tag{30}$$

$$\mathbf{z}_i^{\text{diff}} = \mathcal{G}_{\theta_D}^{\text{Diff}}\left(\mathbf{z}_i^{\text{int}}\right) \in \mathbb{R}^{N_x \times d}, \tag{31}$$

$$\hat{\boldsymbol{\rho}}(t_{i+1}) = \tilde{\boldsymbol{\rho}}(t_i) + \mathcal{P}_{\theta_P}\left(\mathbf{z}_i^{\text{diff}}\right) \in \mathbb{R}^{N_x}, \tag{32}$$

giving the full parametric map:

$$\hat{\mathcal{G}}_\theta\left(\tilde{\boldsymbol{\rho}}(t_i)\right) = \tilde{\boldsymbol{\rho}}(t_i) + \mathcal{P}_{\theta_P}\left(\mathcal{G}_{\theta_D}^{\text{Diff}}\left(\mathcal{G}_{\theta_I}^{\text{Int}}\left(\mathcal{L}_{\theta_L}\left(\tilde{\boldsymbol{\rho}}(t_i)\right)\right)\right)\right). \tag{33}$$

The residual structure encodes the assumption that the field changes smoothly between consecutive steps, so the network learns only the increment rather than the full next state.

Lifting operator $\mathcal{L}_{\theta_L}$. A point-wise linear map $\mathbb{R} \to \mathbb{R}^d$ that independently embeds each scalar $\tilde{\rho}(t_i)(\mathbf{x}_j)$ into a d-dimensional feature vector, yielding $\mathbf{z}_i^{\text{lift}} \in \mathbb{R}^{N_x \times d}$.

Global integral operator $\mathcal{G}_{\theta_I}^{\text{Int}}$. A Transformer encoder of $L_T = 8$ sequentially stacked layers. Denoting $\mathbf{Z}^{(0)} = \mathbf{z}_i^{\text{lift}} \in \mathbb{R}^{N_x \times d}$, each layer $\ell = 1, \ldots, L_T$ applies a pre-norm multi-head self-attention sublayer followed by a pre-norm feed-forward sublayer:

$$\mathbf{A}^{(\ell)} = \text{MultiHead}\left(\text{LN}(\mathbf{Z}^{(\ell-1)})\right) + \mathbf{Z}^{(\ell-1)}, \tag{34}$$

$$\mathbf{Z}^{(\ell)} = \text{FFN}\left(\text{LN}(\mathbf{A}^{(\ell)})\right) + \mathbf{A}^{(\ell)}. \tag{35}$$

With $H = 4$ attention heads and head dimension $d_h = d/H$:

$$\text{MultiHead}(\mathbf{Z}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\, W^O, \qquad \text{head}_h = \text{Attn}\left(\mathbf{Z}W_h^Q,\ \mathbf{Z}W_h^K,\ \mathbf{Z}W_h^V\right), \tag{36}$$

where the scaled dot-product attention discretises the global integral kernel:

$$\mathrm{Attn}(Q,K,V)_i = \sum_{j=1}^{N_x} \underbrace{\mathrm{softmax}\left(\frac{q_i^\top k_j}{\sqrt{d_h}}\right)}_{\kappa_\theta(\mathbf{z}_i,\mathbf{z}_j)} v_j, \tag{37}$$

allowing every spatial point to attend to every other. The feed-forward sublayer uses inner dimension $4d$ and GELU activation:

$$\mathrm{FFN}(\mathbf{a}_i) = W_2\,\mathrm{GELU}(W_1\mathbf{a}_i + b_1) + b_2, \qquad W_1 \in \mathbb{R}^{4d\times d}, \quad W_2 \in \mathbb{R}^{d\times 4d}. \tag{38}$$

The output is $\mathbf{z}_i^{\mathrm{int}} = \mathbf{Z}^{(L_T)} \in \mathbb{R}^{N_x\times d}$.

Local differential operator $\mathcal{G}^{\mathrm{Diff}}_{\theta_D}$. Four residual convolutional blocks acting along the spatial axis. For each block $\ell = 1,\dots,4$, with $\mathbf{h}^{(0)} = \mathbf{z}_i^{\mathrm{int}}$:

$$\mathbf{h}^{(\ell)} = \mathrm{GELU}\big(\mathcal{K}^{(\ell)}_{\theta_D} * \mathbf{h}^{(\ell-1)}\big), \tag{39}$$

$$\mathbf{h}^{(\ell)} \leftarrow \mathrm{LN}\big(\mathbf{h}^{(\ell)}\big) + \mathbf{h}^{(\ell-1)}, \tag{40}$$

where $*$ denotes 1-D convolution along the spatial axis with kernel size $s = 5$, zero-padded by $\lfloor s/2\rfloor = 2$ on each side to preserve the spatial dimension. The kernel $\mathcal{K}^{(\ell)}_{\theta_D} \in \mathbb{R}^{d\times d\times s}$ is zero-sum-constrained at every forward pass:

$$\mathcal{K}^{(\ell)\prime}_{c,c',m} = \mathcal{K}^{(\ell)}_{c,c',m} - \frac{1}{s}\sum_{m'=0}^{s-1}\mathcal{K}^{(\ell)}_{c,c',m'}, \qquad \sum_{m=0}^{s-1}\mathcal{K}^{(\ell)\prime}_{c,c',m} = 0, \tag{41}$$

ensuring each kernel converges to a true differential operator as the grid spacing $h \to 0$. The output is $\mathbf{z}_i^{\mathrm{diff}} = \mathbf{h}^{(4)} \in \mathbb{R}^{N_x\times d}$.

Projection operator $\mathcal{P}_{\theta_P}$. A two-layer point-wise MLP acting independently at each spatial index j, collapsing $\mathbb{R}^d \to \mathbb{R}$:

$$\mathcal{P}_{\theta_P}(\mathbf{z}_j) = W_2\,\mathrm{GELU}(W_1\mathbf{z}_j + b_1) + b_2, \qquad W_1 \in \mathbb{R}^{d\times d}, \quad W_2 \in \mathbb{R}^{1\times d}, \tag{42}$$

with parameters $\theta_P = \{W_1, W_2, b_1, b_2\}$ shared across all j. Applying this map at every j simultaneously yields the increment field $\Delta\hat{\boldsymbol{\rho}} = \mathcal{P}_{\theta_P}(\mathbf{z}_i^{\mathrm{diff}}) \in \mathbb{R}^{N_x}$.

A.3 Training Objective and Evaluation Metrics

DINO is trained by minimising the mean-squared error:

$$\mathcal{L}^{(\mathrm{DINO})}(\theta) = \frac{1}{N_t^{\mathrm{train}}\,N_x}\sum_{i=0}^{N_t^{\mathrm{train}}-2}\sum_{j=1}^{N_x}\Big(\hat{\mathcal{G}}_\theta\big(\tilde{\boldsymbol{\rho}}(t_i)\big)(\mathbf{x}_j) - \tilde{\rho}(t_{i+1})(\mathbf{x}_j)\Big)^2. \tag{43}$$

Performance is assessed primarily through the autoregressive rollout error:

$$\epsilon^{(k)}_{\mathrm{rollout}} = \frac{\|\hat{\rho}^{(k)} - \tilde{\rho}(t_k)\|_1}{\|\tilde{\rho}(t_k)\|_1}, \tag{44}$$

where $\hat{\rho}^{(k+1)} = \hat{\mathcal{G}}_*(\hat{\rho}^{(k)})$ is initialised from $\hat{\rho}^{(0)} = \tilde{\boldsymbol{\rho}}(t_0)$ with no ground-truth injection at subsequent steps.

A.4 Results

Failure Mode 1: Training Convergence. Figure 6 shows training and test loss (Equation 43) on training and test sets over 100 epochs. DINO converges rapidly, reaching $\sim$3–5 $\times 10^{-5}$ with no observable train–test gap throughout training. This apparent success on the one-step task makes the subsequent autoregressive collapse all the more striking.

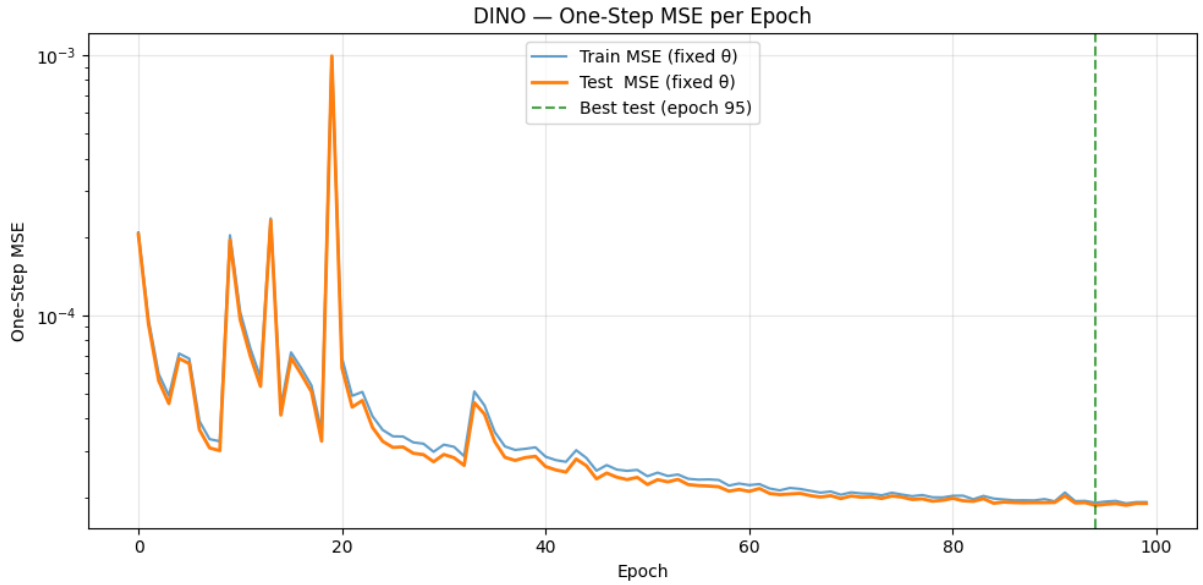


Figure 6: Training and test loss (Equation 43) versus epoch for DINO. Blue line: training loss; Red line: test loss. No train–test gap is observed, yet the model fails catastrophically under autoregressive rollout.

Failure Mode 2: Rollout Error Accumulation (Figure 7). The autoregressive rollout is initialised from the true test profile $\hat{\rho}^{(0)} = \tilde{\rho}(t_0^{\text{test}})$ and advanced recursively as $\hat{\rho}^{(k+1)} = \hat{\mathcal{G}}_\theta(\hat{\rho}^{(k)})$ with no ground-truth injection at subsequent steps. Figure 7 plots the spatially averaged MSE against the ground truth across 1317 rollout steps. Unlike both DeepONet variants, which remain in a bounded oscillatory regime, DINO undergoes rapid error amplification: the spatially averaged MSE rises from $\sim 10^{-4}$ at the seed step to $\sim 10^{1}$ within approximately 50 steps, then plateaus at this elevated level for the remainder of the rollout. This represents a five-order-of-magnitude jump in MSE within a handful of steps, confirming that the residual architecture has no mechanism to prevent compounding errors when applied autoregressively to inputs that drift outside the training distribution. Crucially, this failure occurs under the same checkerboard partition as the DeepONet variants, making it a controlled comparison.

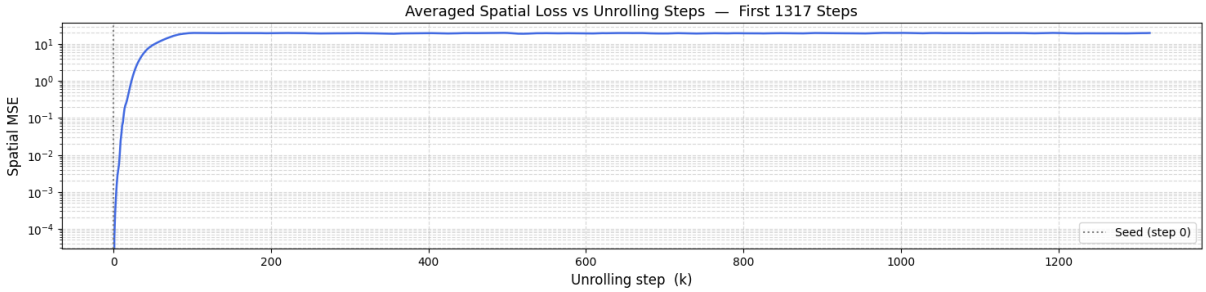


Figure 7: Spatially averaged MSE versus autoregressive rollout step for DINO over 1317 steps (log scale). MSE rises from $\sim 10^{-4}$ at the seed step to $\sim 10^{1}$ within ~ 50 steps, then plateaus at this elevated level, indicating rapid and persistent instability under autoregressive rollout.

Failure Mode 3: Spatiotemporal Prediction Quality (Figures 8–9). Figure 8 shows space–time heatmaps of the ground-truth intensity field alongside the DINO autoregressive prediction and error panels. The ground truth shows the characteristic sparse diagonal striations of the propagating wave envelope. The DINO prediction panel becomes uniformly saturated at the maximum colourscale value almost immediately after rollout begins; the absolute error panel confirms errors reaching ~ 0.30 across the entire domain within the first few steps, and the relative error panel is essentially 1.0 everywhere beyond the first step. In sharp contrast, Swin-DeepONet partially recovers these striations, as shown in Section 6.

Figure 9 traces the per-step spatial profile progression. At steps 1 and 2, DINO tracks the ground truth closely with only minor deviations at the sharp peaks. By step 3, the prediction still follows the broad envelope but exhibits small spurious oscillations. By step 10, the prediction has fully diverged: it displays dense high-frequency noise, crosses zero to unphysical negative values, and bears no structural resemblance to the true field. From step 25 onward, the prediction is locked to a uniformly elevated, high-frequency noisy state at approximately 0.08–0.30, entirely divorced from a ground truth that remains near zero with sparse sharp peaks. This locked state persists without any self-correction. The root cause is that each residual update accumulates error and shifts the input distribution away from the training regime, triggering a positive feedback loop that the model cannot escape.

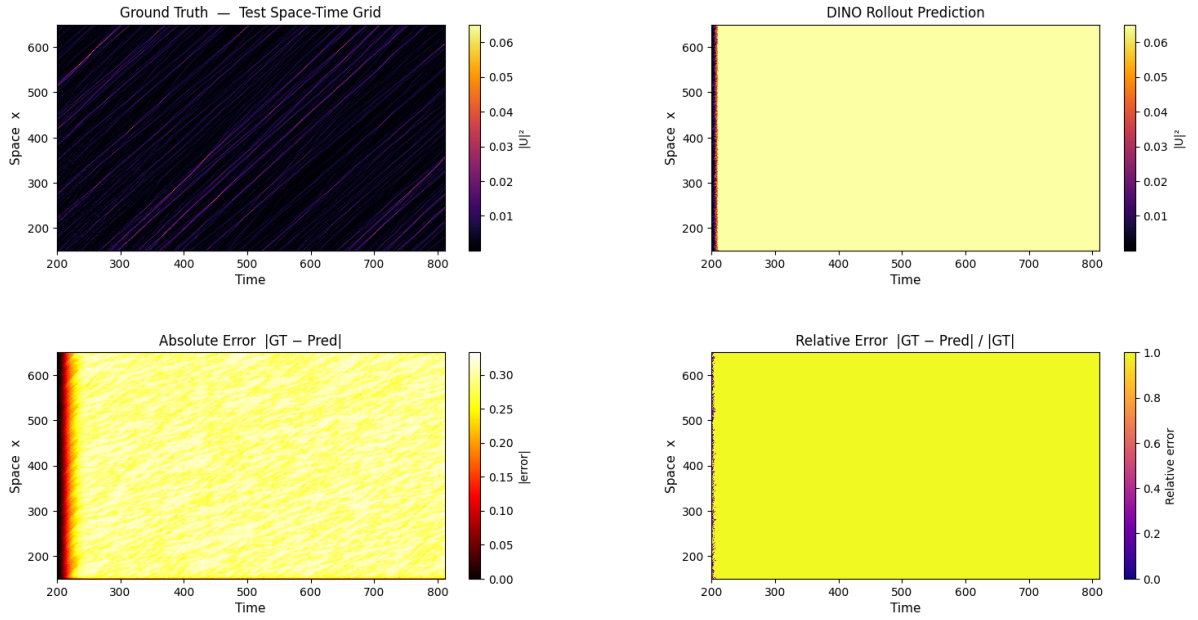
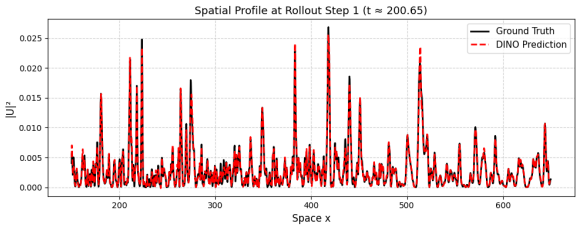
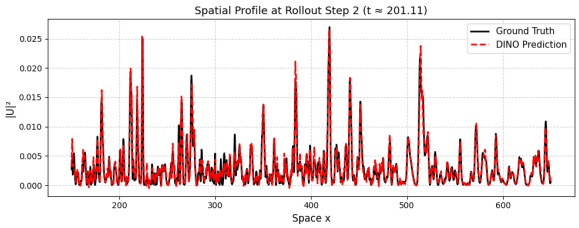


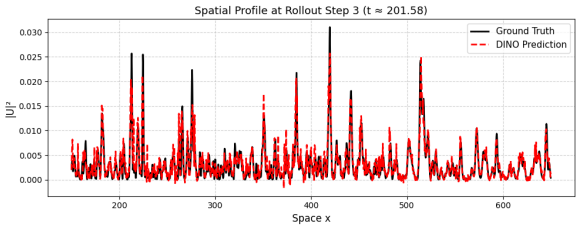
Figure 8: Space–time heatmaps for DINO over the full 1317-step rollout. *Top left:* ground-truth intensity $\tilde{\rho}(t_k, x_j)$ showing sparse diagonal propagating structures. *Top right:* DINO autoregressive prediction, which saturates to a uniformly high value within the first few steps. *Bottom left:* absolute pointwise error $|\hat{\rho}^{(k)} - \tilde{\rho}|$, reaching ~0.30 across the full domain. *Bottom right:* relative pointwise error, essentially 1.0 everywhere after the initial steps.



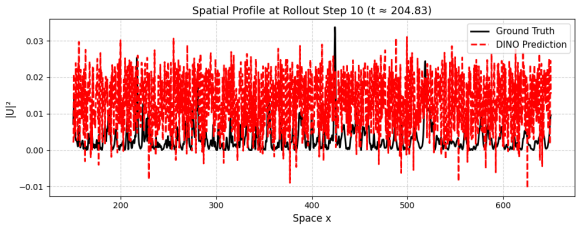
(a) Step 1 — accurate tracking



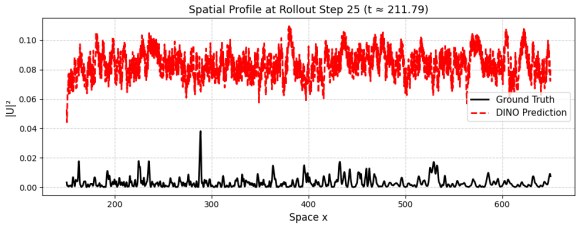
(b) Step 2 — still closely tracking



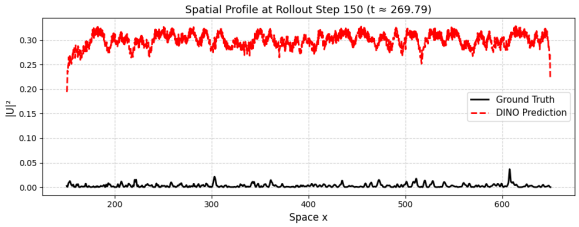
(c) Step 3 — onset of spurious oscillations



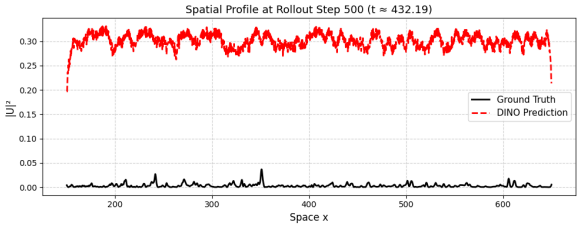
(d) Step 10 — full divergence, unphysical negative values



(e) Step 25 — locked to saturated high-amplitude state



(f) Step 150 — persistent saturation, no self-correction



(g) Step 500 — unchanged saturation after 500 steps

Figure 9: DINO autoregressive failure progression. Steps 1–2 show accurate prediction; step 3 shows the onset of spurious high-frequency oscillations; step 10 shows full divergence with unphysical negative intensities; steps 25–500 show the model locked to a uniformly elevated noisy state with no resemblance to the ground truth.