SIMULATING FLUID DYNAMICS WITH A PARAMETER-EFFICIENT NEURAL ARCHITECTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural operators provide fast surrogates for fluid forecasting, but accurate prediction with a compact model remains challenging. Fluid evolution combines spatial transport with changes in field values, yet many architectures represent both through the same latent transformations, which can increase the representational burden. To address this, we introduce SIDDY, a parameter-efficient neural architecture that reuses observed content through explicit transport and learns a refinement process for the remaining evolution. Specifically, a shared encoder first summarizes the full observation history to guide both transport and refinement, while the latest field supplies the state that is explicitly evolved. Semi-Lagrangian sampling then transports this field once, after which reaction-diffusion layers refine the transported state through learned nonlinear updates and fixed Laplacian stencils. This physics-inspired decomposition enables accurate forecasting with a compact parameterization. Experiments on both simulated and real-world dynamics show that SIDDY achieves the lowest L_2 error across all datasets while using only a fraction of the parameters.

1 INTRODUCTION

Applications such as engineering simulation (Li et al., 2023c; Pfaff et al., 2021) and weather forecasting (Bodnar et al., 2025; Bi et al., 2023; Lam et al., 2023) rely on repeated predictions of fluid evolution. Because high-resolution numerical simulation is expensive, neural operators (Kovachki et al., 2023; Lu et al., 2021) have become effective surrogates that can be evaluated repeatedly after training. A central challenge is to maintain forecast accuracy while keeping the model compact. Existing approaches improve spatial modeling through different mechanisms, including Fourier layers (Li et al., 2021), attention mechanisms (Wu et al., 2024), and physics-inspired architectures (Zakariaei et al., 2024; Pereira et al., 2026). However, many of these approaches rely on latent feature transformations to represent the entire evolution process. As a result, the same parameterized operations are often required to describe both the movement of existing structures and the generation of new field variations. These two processes involve different types of information, which motivates a more explicit separation of their roles.

Fluid dynamics provides a natural perspective for such a separation. Advection primarily transports existing field content, while diffusion and local interactions modify the transported state. Instead of reconstructing the complete future field through latent transformations, observed structures can be directly carried forward through learned spatial transport. The remaining evolution can then be modeled as a refinement process that captures changes beyond transport. This perspective leads to a compact architecture that assigns explicit transport to observed content and allocates additional learning capacity to refinement.

The same principle also applies to temporal information. Previous observations contain important cues about motion and field evolution, but these observations do not need to be explicitly transported together with the current state. A more efficient strategy is to extract temporal information into a shared context representation and use this context to guide both transport and refinement. Meanwhile, the latest observed field provides the physical content that is propagated forward. Such a separation allows the model to exploit the full observation history while avoiding unnecessary repeated sampling of historical states.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Following these principles, we introduce SIDDY (**Si**mulating Flui**d Dy**namics), a parameter-efficient neural architecture for fluid forecasting. A shared encoder first summarizes the full observation history, producing a context that guides both transport and refinement. Conditioned on this context, a semi-Lagrangian-inspired sampling stage combines values from learned source locations in the latest observed field to construct a transported state. The transported field is then lifted into feature space and refined by L reaction-diffusion layers, which combine learned nonlinear updates with fixed Laplacian stencils under the same context. The resulting features are decoded into a refinement that is added to the transported prediction to obtain the final forecast. Empirically, SIDDY consistently achieves the lowest L_2 error in various scenarios with a substantially smaller model size.

Our contributions are summarized as follows:

- We propose SIDDY, a parameter-efficient architecture that explicitly transports observed fields and allocates iterative computation to reaction-diffusion refinement. This design reuses existing field content through transport while reserving model capacity for the remaining evolution.
- We develop a shared history encoder that conditions both transport and refinement. This allows SIDDY to exploit full observation histories while restricting explicit transport to the latest field.
- We evaluate SIDDY on nine settings covering simulated and real-world dynamics. SIDDY achieves the lowest L_2 error with fewer parameters than the baselines.

2 Related Work

Representation-based neural operators differ in how they parameterize spatial interactions (Rahman et al., 2023; Bonev et al., 2023; Raonic et al., 2023). FNO and related Fourier-based operators (Li et al., 2021; Guibas et al., 2022; Tran et al., 2023; Li et al., 2023b) model these interactions through learned transformations in Fourier space. To capture localized and multiscale structure, WNO (Tripura & Chakraborty, 2023) uses wavelets for spatial and frequency localization, while M2NO (Li et al., 2026) combines multiwavelets with multigrid transfers to connect coarse and fine scales. Other approaches focus on transient responses or position-dependent operators. LNO (Cao et al., 2024) represents transient responses through poles and residues, whereas KANO (Lee et al., 2026) combines spatial and spectral bases for position-dependent operators. These methods improve the representation of spatial interactions, but transport is generally left to the learned latent feature transformations themselves. SIDDY instead moves observed physical fields explicitly before refining the transported state.

Attention-based operators learn input-dependent weights for nonlocal interactions (Calvello et al., 2025; Li et al., 2023a; Cao, 2021; Hao et al., 2023). Because full spatial attention has quadratic cost, Transolver (Wu et al., 2024) aggregates features into learned slices, while LRSA (Yang et al., 2026) exchanges information through a small set of latent tokens. SIDDY also computes attention on a coarse grid, but uses it only to build the history context and combines it with local features at the original resolution. The context is then shared by transport and refinement rather than serving as the forecast state itself.

Physical knowledge provides another way to structure learning through training objectives or architectural design (Raissi et al., 2019; Um et al., 2020; Long et al., 2018; Le Guen & Thome, 2020). For training objectives, PINO (Li et al., 2024) supplements data supervision with equation residuals, which must be evaluable for the modeled variables. For architecture design, localized integral and differential kernels (Liu-Schiaffini et al., 2024) complement global interactions with fine-scale structure. HelmFluid (Xing et al., 2024) uses potential and stream functions for motion modeling, while Flowers (Muser et al., 2026) learns multihead feature warping. These methods show that explicit sampling can provide a useful alternative to representing motion only through feature mixing.

A related line of research combines transport with other physics-inspired updates. ADRNet (Zakariaei et al., 2024) and PARADIS (Pereira et al., 2026) repeatedly update latent states through operator splitting, allowing transport and feature changes to interact throughout depth. HyCOP (Zhao et al., 2026) learns which numerical or learned sub-solvers to select from a dictionary and how long to apply them. SIDDY instead uses a different organization. Observed physical fields are transported

explicitly once, followed by repeated reaction-diffusion refinement, while the shared encoding of the full history guides both steps.

3 SIDDY ARCHITECTURE

As shown in Figure 1, SIDDY combines a shared history encoder, explicit transport of observed fields, and L refinement layers. Let $u_n \colon \Omega_h \to \mathbb{R}^C$ be a field on a two-dimensional grid, and let $U_n = (u_{n-T+1}, \ldots, u_n)$ contain T observations at a fixed interval. With $U_n^{(S)} = (u_{n-S+1}, \ldots, u_n)$ denoting the S most recent fields for $1 \le S \le T$, the prediction is given by

$$\begin{aligned} c_n &= \mathcal{E}_\theta(U_n), \\ \widetilde{u}_{n+1} &= \mathcal{T}_\theta(U_n^{(S)}; c_n), \\ \widehat{u}_{n+1} &= \mathcal{B}_\theta(\widetilde{u}_{n+1}) + \mathcal{P}_\theta(h^{(L)}). \end{aligned} \tag{1}$$

The context c_n encodes all T observations and conditions both transport and refinement, while $U_n^{(S)}$ supplies the fields sampled during transport. The transport stage produces $\widetilde{u}_{n+1}$, which is lifted into features used to initialize $h^{(0)}$. Specifically, transport is evaluated once per prediction step, after which the refinement layers update the transported features. The map $\mathcal{B}_\theta$ projects the transported physical field to the output variables, while $\mathcal{P}_\theta$ decodes the refined features into a correction.

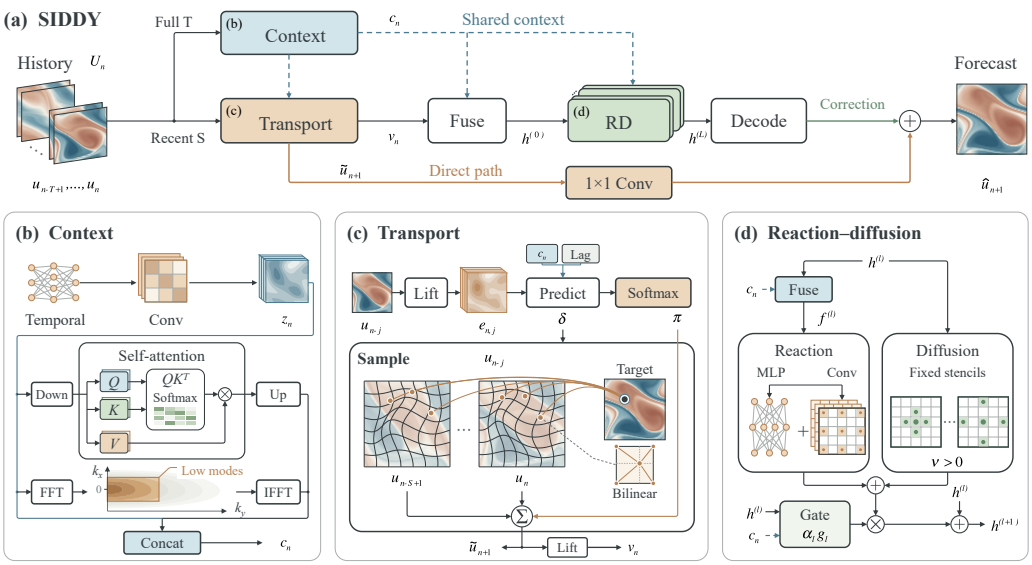

Figure 1: **Overview of SIDDY.** A shared history context guides both transport and refinement. Semi-Lagrangian-inspired sampling moves selected observed fields toward the forecast locations. The following L reaction-diffusion layers then update the transported state.

3.1 SPATIOTEMPORAL CONTEXT

The encoder first extracts temporal information within each physical variable, then combines the resulting features across variables and nearby locations through

$$\begin{aligned} q_n^{(m)}(x) &= \tau_{\theta,m}\left([u_{n-T+1}^{(m)}(x), \ldots, u_n^{(m)}(x)]\right), \\ z_n &= \mathcal{H}_\theta\left([q_n^{(1)}, \ldots, q_n^{(C)}]\right). \end{aligned} \tag{2}$$

Each $\tau_{\theta,m} \colon \mathbb{R}^T \to \mathbb{R}^p$ is a two-layer pointwise network shared across spatial locations. Grouped channel operations keep temporal mixing within each variable, after which $\mathcal{H}_\theta$ combines their outputs through pointwise projections and multiscale spatial convolutions. The resulting d-channel field z_n therefore contains both temporal and local spatial information.

Motion and field changes can also depend on flow outside these local neighborhoods, so we supplement z_n with attention and, for periodic problems, optional spectral features. The two global branches and their fusion are defined by

$$\begin{aligned} g_n^{\mathrm{att}} &= \mathrm{Up}_s(\mathcal{A}_\theta(\mathrm{Down}_s(z_n)))\,, \\ g_n^{\mathrm{sp}} &= \mathcal{F}^{-1}(M_\theta \mathcal{F} z_n)\,, \\ c_n &= \Phi_\theta\big([z_n, g_n^{\mathrm{att}}, g_n^{\mathrm{sp}}]\big)\,. \end{aligned} \tag{3}$$

To limit the cost of spatial self-attention (Vaswani et al., 2017), the first branch reduces resolution through convolution and average pooling before applying attention, then restores the result by bilinear interpolation. Reducing each dimension of an N-point grid by s gives approximately N/s^2 tokens and a pairwise attention cost of $O(N^2 d/s^4)$. The parallel spectral branch applies Fourier mixing (Li et al., 2021) to retained modes of the original-resolution features. Fusing these outputs with z_n combines both global and local interactions. The resulting context field is reused by both transport and refinement.

3.2 Transport of Observed Fields

Given c_n, SIDDY transports the selected observed fields with differentiable multi-candidate sampling inspired by semi-Lagrangian advection (de Bézenac et al., 2018; Staniforth & Côté, 1991). For each field u_{n-j}, a shared network takes its lifted features $e_{n,j}$, c_n, and a lag embedding to predict K backward displacements $\boldsymbol{\delta}_{n,j,k}$ and scores $a_{n,j,k}$. Normalizing the scores jointly across frames and candidates gives a weighted combination of source values at each target point x,

$$\begin{aligned} \pi_{n,j,k}(x) &= \frac{\exp(a_{n,j,k}(x))}{\sum_{j'=0}^{S-1}\sum_{k'=1}^{K}\exp(a_{n,j',k'}(x))}, \\ \widetilde{u}_{n+1}(x) &= \sum_{j=0}^{S-1}\sum_{k=1}^{K}\pi_{n,j,k}(x)\,\mathcal{I}[u_{n-j}](x+\boldsymbol{\delta}_{n,j,k}(x))\,. \end{aligned} \tag{4}$$

Here x and $\boldsymbol{\delta}$ use normalized grid coordinates, $\mathcal{I}$ denotes bilinear interpolation, and the scores include a learned frame bias. All physical channels share sampling locations and weights, so their values come from matching source points. The window S determines which fields provide source values, while the full history conditions their locations and weights through c_n. With $S = 1$, sampling uses the latest field and the encoder retains all T observations.

The transported physical field is lifted to initialize the refinement layers,

$$v_n = \mathcal{L}_{\mathrm{post}}(\widetilde{u}_{n+1}). \tag{5}$$

Appendix A describes the lifting network, displacement parameterization, and boundary handling.

3.3 Reaction and Diffusion Refinement

After transport, the refinement layers model changes in field values and local structure. We first combine the transported features with the history context through $h^{(0)} = v_n + W_{\mathrm{in}}[v_n, c_n]$, where W_{in} is pointwise affine. Each layer then updates the current state using the same context through

$$\begin{aligned} f^{(\ell)} &= h^{(\ell)} + V_\ell[h^{(\ell)}, c_n], \\ \mathcal{R}_\ell(h^{(\ell)}; c_n) &= W_{\ell,2}\,\sigma\Big(W_{\ell,1}\,\mathrm{Norm}(f^{(\ell)})\Big) + \mathcal{S}_\ell(f^{(\ell)}). \end{aligned} \tag{6}$$

The pointwise maps $W_{\ell,1}$ and $W_{\ell,2}$ mix channels through GELU activation σ. The multiscale spatial mixer $\mathcal{S}_\ell$ adds an increment that incorporates neighboring features into the update.

A diffusion branch complements the learned update with a prescribed pattern of spatial exchange,

$$\mathcal{D}_\ell h = \sum_{r\in\mathscr{S}} \boldsymbol{\nu}_{\ell,r} \odot \Delta_r h, \qquad \boldsymbol{\nu}_{\ell,r} > 0. \tag{7}$$

The radius-r operator uses the five-point stencil

$$(\Delta_r h)(x) = \sum_{i=1}^{2}[h(x + r\boldsymbol{e}_i) + h(x - r\boldsymbol{e}_i)] - 4h(x), \tag{8}$$

where offsets are measured in grid cells and $\boldsymbol{\nu}_{\ell,r} \in \mathbb{R}^d$ contains positive channel-wise strengths parameterized by softplus. Combining radii extends exchange across scales while learning only $d|\mathscr{S}|$ coefficients per layer. Unlike $\mathcal{S}_\ell$, which learns responses to neighboring features and context, this branch fixes the spatial difference pattern and adapts its strength separately for every feature channel and each of the selected radii.

The two contributions are combined in a gated update of the current feature state,

$$h^{(\ell+1)} = h^{(\ell)} + \boldsymbol{\alpha}_\ell \odot \boldsymbol{g}_\ell(h^{(\ell)}, c_n) \odot \left[\mathcal{R}_\ell(h^{(\ell)}; c_n) + \mathcal{D}_\ell h^{(\ell)}\right]. \tag{9}$$

The sigmoid gate $\boldsymbol{g}_\ell$ adapts the combined increment to each location and channel, while $\boldsymbol{\alpha}_\ell \in \mathbb{R}^d$ provides a learned layer scale. Since each updated state is passed to the next layer, the learned response and diffusion branch interact over successive refinements. Increasing L adds refinement capacity while reusing the same context and transported features.

Finally, Equation 1 combines the prediction obtained directly from the transported field with the decoded refinement features. Since the final projection of $\mathcal{P}_\theta$ is zero-initialized, training starts near the transport-only prediction and gradually learns the correction term. Algorithm 1 in Appendix A summarizes one prediction step, which is applied recursively by appending the predicted field to the history and removing its oldest frame. The same appendix details the network blocks, initialization, default settings, and computational cost.

4 Experiments

We evaluate SIDDY on simulated fluid dynamics, observed ink dispersion, and global atmospheric forecasting. Navier-Stokes provides controlled vorticity dynamics with different balances between transport and viscous smoothing. Real and synthetic ink videos test long-horizon extrapolation of moving structures and image detail from RGB observations, spanning recorded and simulated fluid motion. ERA5-Z500 extends the evaluation to global atmospheric dynamics represented by reanalysis data. The main comparisons measure prediction accuracy together with model size across these settings. We then conduct ablation studies on Navier-Stokes to examine how each architectural component affects forecasting performance. Throughout this paper, L_2 denotes relative L_2 error, and we report the mean values of three independent runs.

4.1 Experimental Setup

Datasets and baseline configurations. Appendix B summarizes the data sources, partitions, and forecasting windows. We compare SIDDY with representation-based architectures FNO (Li et al., 2021) and M2NO (Li et al., 2026), the attention-based architecture LRSA (Yang et al., 2026), and physics-guided architectures ADRNet (Zakariaei et al., 2024), HelmFluid (Xing et al., 2024), and Flowers (Muser et al., 2026). Model and training settings are given in Appendices C and D. All experiments are conducted on a single 32G graphics card, except for the LRSA on ERA5-Z500, for which we used two graphics cards.

Evaluation. On Navier-Stokes, we report L_2 error over the complete rollout. For Spread Ink, we report L_2, mean squared error (MSE), and VGG perceptual loss (Johnson et al., 2016), which compares pretrained image features and complements errors measured directly in field values. For ERA5-Z500, we report L_2 and MSE. All main-table errors are aggregated over the complete forecast horizons (Appendix D.1), and lower values indicate better predictions. Parameter counts (#P) are reported in millions (M). SIDDY's computational cost and parameter accounting are detailed in Appendix A.4. Percentage changes are computed as $\Delta = 100(v/v_{\mathrm{ref}} - 1)$. Table 1 reports changes below each baseline using that method as v_{ref}. Table 2 references the best baseline per metric or the smallest baseline parameter count. **Bold** and underline mark the best and second-best main-table values, while **bold** marks the lowest values within each ablation group. Changes are shown by magnitude, with green ↓ for decreases and red ↑ for increases.

4.2 Main Results

Navier-Stokes and ERA5-Z500. Table 1 compares prediction accuracy and model size on simulated fluid dynamics and observed atmospheric fields. On Navier-Stokes, SIDDY reduces the L_2

error over the rollout by 25.2 to 57.0% compared with the strongest baseline at each viscosity, while using 18.6% fewer parameters than FNO, the smallest baseline in this family. The same trend appears across all three viscosity settings. Figure 2 shows forecasts and error maps. Appendix F reports physical diagnostics. On ERA5-Z500, SIDDY also achieves the lowest errors, reducing L_2 error and MSE by 3.2% and 6.8% relative to ADRNet, while the number of parameters was significantly reduced by 90.4%. FNO is the smallest baseline with 5.317M parameters, while SIDDY uses 2.498M, a 53.0% decrease, and obtains lower errors. These results demonstrate simultaneous improvements in accuracy and parameter efficiency on both simulated and observed fields. Together, SIDDY consistently uses fewer parameters than the smallest baseline in each family while attaining lower forecast errors.

Table 1: **Navier-Stokes and ERA5-Z500 forecasting.** The three Navier-Stokes L_2 columns correspond to $\nu = 10^{-3}$, 10^{-4}, and 10^{-5} from left to right.

Method	Navier-Stokes				ERA5-Z500		
	#P	L_2			#P	L_2	MSE
FNO	<u>1.19</u>	0.0128	0.1559	0.1556	<u>5.32</u>	0.7649	1035191
Δ	↓18.6%	↓73.4%	↓84.5%	↓77.1%	↓53.0%	↓63.1%	↓87.8%
ADRNet	9.44	0.0137	0.1768	0.1770	26.11	<u>0.2917</u>	<u>135888</u>
Δ	↓89.7%	↓75.2%	↓86.4%	↓79.8%	↓90.4%	↓3.2%	↓6.8%
HelmFluid	9.93	0.1589	1.1615	0.1644	9.81	0.5089	413535
Δ	↓90.3%	↓97.9%	↓97.9%	↓78.3%	↓74.5%	↓44.5%	↓69.4%
M2NO	17.34	0.0099	0.0614	<u>0.0655</u>	9.82	0.4736	355392
Δ	↓94.4%	↓65.7%	↓60.7%	↓45.5%	↓74.6%	↓40.4%	↓64.4%
Flowers	17.33	0.5884	0.1207	0.1233	17.33	0.2977	140591
Δ	↓94.4%	↓99.4%	↓80.0%	↓71.0%	↓85.6%	↓5.2%	↓9.9%
LRSA	14.55	<u>0.0079</u>	<u>0.0322</u>	0.0804	9.15	0.5970	567320
Δ	↓93.3%	↓57.0%	↓25.2%	↓55.6%	↓72.7%	↓52.7%	↓77.7%
SIDDY	**0.97**	**0.0034**	**0.0241**	**0.0357**	**2.50**	**0.2823**	**126617**

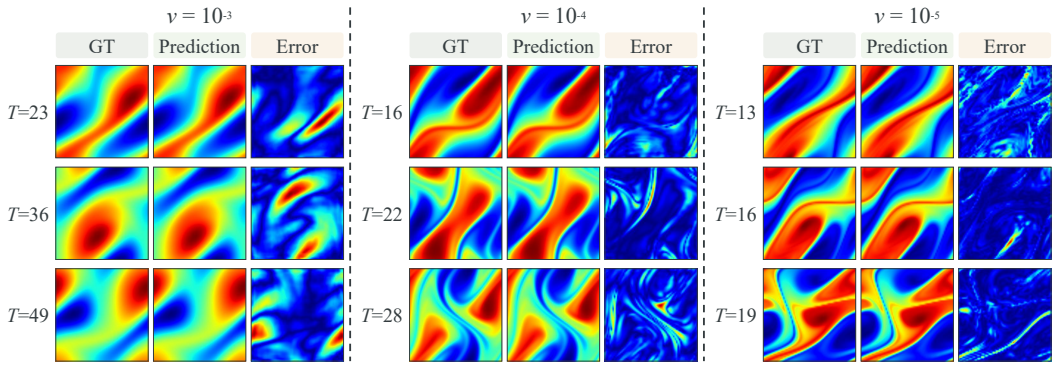

Figure 2: **Navier-Stokes forecasts from SIDDY.** Each viscosity group shows ground truth, prediction, and error from left to right. The time labels refer to the recorded trajectories.

Spread Ink. Table 2 compares Spread Ink forecasts, where numerical and perceptual metrics assess different aspects of the forecast. We remove the spectral branch from the history encoder because the problem does not involve periodic boundaries. SIDDY achieves the lowest L_2 and MSE on all five videos while using 85.9% fewer parameters than HelmFluid, the smallest baseline in this family. On the three real sequences, it also achieves the lowest VGG loss. For Real 1 and 2, the VGG reductions of 16.2% and 16.9% relative to HelmFluid show that the improvement also appears in pretrained image features. On Real 3, Flowers is the strongest baseline for all three metrics, and SIDDY reduces L_2, MSE, and VGG loss by 3.8%, 7.6%, and 2.9%, respectively. While these improvements are more modest than those on the previous two datasets, SIDDY's advantages remain

Table 2: **Spread Ink forecasting.**

Dataset	Metric	FNO	ADRNet	HelmFluid	M2NO	Flowers	LRSA	**SIDDY**	Δ (%)
#P (M)		18.91	28.82	10.68	58.46	17.33	14.56	**1.50**	↓85.9
Real 1	L_2	0.1702	4.5782	0.1382	0.1573	0.1729	0.2033	**0.1258**	↓9.0
	MSE	0.0096	6.9981	0.0064	0.0083	0.0100	0.0138	**0.0053**	↓17.2
	VGG	2.0245	430.2042	1.5310	1.6738	1.8150	2.8610	**1.2824**	↓16.2
Real 2	L_2	0.5134	3.9925	0.3654	0.4827	0.4601	0.5112	**0.3164**	↓13.4
	MSE	0.0381	2.0564	0.0193	0.0337	0.0307	0.0378	**0.0145**	↓24.9
	VGG	5.2413	168.9067	3.4523	4.1718	4.1038	6.5266	**2.8682**	↓16.9
Real 3	L_2	0.1617	2.1539	0.1644	0.1692	0.1614	0.1662	**0.1553**	↓3.8
	MSE	0.0133	2.3586	0.0137	0.0145	0.0132	0.0140	**0.0122**	↓7.6
	VGG	5.3882	130.2798	5.4363	5.7374	5.3642	6.0740	**5.2109**	↓2.9
Synthetic 1	L_2	0.4540	3.0621	0.9861	0.4056	0.3636	0.4674	**0.3569**	↓1.8
	MSE	0.0638	2.9011	0.3008	0.0509	0.0409	0.0676	**0.0394**	↓3.7
	VGG	23.6351	274.1992	13.4053	12.6233	**10.3520**	16.9477	11.0395	↑6.6
Synthetic 2	L_2	0.3449	1.7237	1.0339	0.3651	0.3478	0.3404	**0.2891**	↓15.1
	MSE	0.0810	2.0227	0.7278	0.0907	0.0823	0.0789	**0.0569**	↓27.9
	VGG	13.8616	123.2834	15.4880	12.9042	12.7983	21.7128	**11.8296**	↓7.6

evident in terms of both model size and prediction error. Figure 3 provides the corresponding visual comparison on Real 1. On Synthetic 1, SIDDY achieves the lowest L_2 and MSE, while Flowers achieves the lowest VGG loss. Appendix G compares the per-step errors over the 200-step rollout. On Synthetic 2, SIDDY leads all three metrics, reducing L_2 and MSE by 15.1% and 27.9% relative to LRSA and VGG loss by 7.6% relative to Flowers. SIDDY therefore obtains the lowest VGG loss on four of the five videos and the lowest L_2 and MSE on all five. Additional visual comparisons for the remaining videos and ERA5-Z500 are provided in Appendix H.

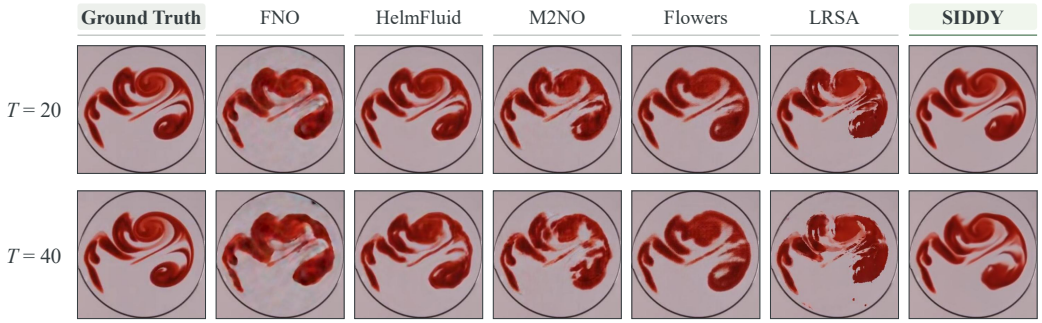

Figure 3: **Prediction results on Spread Ink Real 1.** Columns show ground truth and predictions from the indicated methods at prediction steps 20 and 40.

Across the two main tables, although the strongest baseline varies by task, SIDDY consistently obtains the lowest L_2 error on all nine evaluated settings. We next examine the contributions of history conditioning, sampling, refinement depth, and stage organization.

4.3 Ablation Studies

Table 3 examines the model components and their organization in ablation studies. The transport window S sets the number of recent fields available for sampling, K is the number of candidate source locations per field, and L is the number of reaction-diffusion refinement layers. All variants

with a context encoder use the full history of T observations. The sweep over S changes the fields used for transport while keeping the encoder input fixed.

Table 3: **Component and architecture ablations on Navier-Stokes.** In each pair, the first row reports the original model and the aligned row a variant with a parameter count comparable to SIDDY. Both w/o context variants transport all ten history frames. The original w/o context variant is the reference for the first group. Other groups use SIDDY ($S = 1$, $K = 2$, $L = 4$) as the reference. An asterisk marks the default setting.

Setting	$\nu = 10^{-3}$		$\nu = 10^{-4}$		$\nu = 10^{-5}$		Parameters	
	L_2	Δ	L_2	Δ	L_2	Δ	#P (M)	Δ
Context and transported history								
w/o context	0.0205	—	0.1261	—	0.1283	—	0.455	—
aligned	0.0337	↑64.4%	0.0970	↓23.1%	0.1074	↓16.3%	1.015	↑123.12%
$S = 1^*$	**0.0034**	↓83.4%	**0.0241**	↓80.9%	**0.0357**	↓72.2%	0.968	↑112.82%
$S = 4$	0.0043	↓79.0%	0.0259	↓79.5%	0.0388	↓69.8%	0.968	↑112.86%
$S = 7$	0.0042	↓79.5%	0.0244	↓80.7%	0.0367	↓71.4%	0.968	↑112.90%
$S = 10$	0.0042	↓79.5%	0.0250	↓80.2%	0.0359	↓72.0%	0.968	↑112.95%
Sampling candidates								
$K = 1$	0.0038	↑11.8%	0.0247	↑2.5%	0.0341	↓4.5%	0.968	↓0.02%
$K = 2^*$	**0.0034**	—	0.0241	—	0.0357	—	0.968	—
$K = 4$	0.0036	↑5.9%	0.0235	↓2.5%	**0.0335**	↓6.2%	0.968	↑0.04%
$K = 8$	0.0037	↑8.8%	**0.0222**	↓7.9%	0.0339	↓5.0%	0.969	↑0.12%
Correction depth								
$L = 0$	0.0036	↑5.9%	0.0276	↑14.5%	0.0389	↑9.0%	0.900	↓7.01%
$L = 2$	0.0046	↑35.3%	0.0248	↑2.9%	0.0365	↑2.2%	0.934	↓3.50%
$L = 4^*$	**0.0034**	—	0.0241	—	0.0357	—	0.968	—
$L = 6$	0.0048	↑41.2%	**0.0223**	↓7.5%	**0.0338**	↓5.3%	1.002	↑3.50%
Architectural organization								
SIDDY*	0.0034	—	0.0241	—	0.0357	—	0.968	—
Refinement first	0.0034	0.0%	0.0262	↑8.7%	0.0349	↓2.2%	0.968	0.00%
Refinement only	0.0059	↑73.5%	0.0482	↑100.0%	0.0688	↑92.7%	0.696	↓28.09%
aligned	0.0046	↑35.3%	0.0430	↑78.4%	0.0621	↑73.9%	1.084	↑11.96%
Interleaved stages	**0.0027**	↓20.6%	0.0196	↓18.7%	**0.0257**	↓28.0%	1.793	↑85.25%
Repeated transport	0.0030	↓11.8%	**0.0182**	↓24.5%	0.0293	↓17.9%	1.621	↑67.52%

Context and transported history. At $S = 10$, adding the context encoder reduces errors by 79.5%, 80.2%, and 72.0% relative to the original no-context variant. To compare at similar parameter counts, we widen the no-context model to 1,014,703 parameters, 4.8% more than the model with context at the same $S = 10$. Widening reduces the no-context errors by 23.1% and 16.3% at $\nu = 10^{-4}$ and 10^{-5}, but increases the error by 64.4% at $\nu = 10^{-3}$. The model with shared context achieves 87.5%, 74.2%, and 66.6% lower errors than this widened control. Shared history conditioning therefore provides larger gains than widening the no-context model. Sampling only the latest field gives the lowest error at all three viscosities. We therefore set $S = 1$, using the full history for conditioning and the latest field for transport. At fixed K, this reduces candidate sampling tenfold relative to $S = 10$, whereas the parameter count is nearly unchanged.

Sampling candidates. Increasing K gives each target point more possible source locations. The lowest errors occur at $K = 2$, 8, and 4 for the three viscosities, respectively. We use $K = 2$ to keep the number of sampled candidates small. Increasing K from two to eight adds only 0.12% parameters but quadruples candidate sampling and increases computation and intermediate storage.

Refinement depth. The $L = 0$ variant removes the iterative refinement layers. At $\nu = 10^{-4}$ and 10^{-5}, increasing L from zero to six progressively reduces error by 19.2% and 13.1%, with an 11.3% increase in parameters. These gains come from additional refinement layers, with the context encoder and transport stage unchanged. At $\nu = 10^{-3}$, error decreases from 0.0036 at $L = 0$ to 0.0034 at $L = 4$, then rises to 0.0048 at $L = 6$. The non-monotonic behavior suggests that the optimal refinement depth depends on the dynamics of the forecasting task. Each layer combines learned updates, spatial mixing, and gated diffusion, so increasing depth is not equivalent to increasing physical diffusion, but controls the capacity of the full refinement block.

Transport and refinement order. The last group in Table 3 compares stage organizations. SIDDY applies one transport stage followed by four refinement layers, while the refinement-first variant reverses this order. At the same parameter count, reversing the order leaves the error unchanged at $\nu = 10^{-3}$, increases it by 8.7% at $\nu = 10^{-4}$, and decreases it by 2.2% at $\nu = 10^{-5}$. The preferred order therefore varies across the three settings. In SIDDY, we place transport first to carry observed content to the forecast locations, so that refinement operates on the transported state and learns the remaining correction. This gives the transported prediction a direct role as the basis for the final forecast, consistent with the residual formulation in Equation 1.

Removing transport. Complementing $L = 0$, which removes reaction-diffusion refinement, the original refinement only variant removes transport while retaining four refinement layers. Relative to SIDDY, this variant uses 28.1% fewer parameters and increases errors by 73.5%, 100.0%, and 92.7%. Increasing its parameter count reduces errors to some extent. The larger variant has 11.96% more parameters than SIDDY but errors that are 35.3%, 78.4%, and 73.9% higher. Removing transport therefore has a larger effect on error than removing refinement at all three viscosities. This comparison highlights the value of explicitly reusing observed field content rather than relying only on learned feature updates. Both variants outperform several baselines at all three viscosities (Table 1), showing that history-conditioned refinement remains effective on its own.

Interleaving transport and refinement. The interleaved variant stacks four blocks, each containing one transport stage and one refinement layer. Relative to SIDDY, it reduces errors by 18.7% to 28.0% while increasing the parameter count to 1.793M, an 85.2% increase. This repeated organization resembles ADRNet (Zakariaei et al., 2024), which evolves a joint latent representation of the input window. The interleaved SIDDY variant retains the separation of shared history conditioning from state evolution and achieves lower errors than ADRNet at all three viscosities with 81.0% fewer parameters (Table 1). The gains in accuracy and model size therefore extend to the interleaved organization. By comparison, increasing refinement depth from four to six adds only 3.5% parameters and reduces errors at the two lower viscosities without extra spatial sampling.

Repeating transport before refinement. The repeated-transport variant applies four transport stages followed by one refinement layer, reversing SIDDY's allocation of depth. Errors decrease by 11.8% to 24.5%, while parameters grow to 1.621M, a 67.5% increase. Compared with interleaving, this variant achieves a lower error at $\nu = 10^{-4}$, whereas interleaving is more accurate at the other two viscosities. Neither expanded organization is uniformly more accurate than the other, and both require substantially more parameters than the default setting. The single-transport design therefore provides a compact default, while repeated transport offers a higher-capacity alternative.

The ablations show that shared history conditioning improves forecasting while transport samples only the latest field. Deeper refinement adds capacity without repeated sampling, while more transport stages trade model size for lower errors. For additional ablation results, see Appendix E.

5 Conclusion

We introduced SIDDY, a compact architecture that separates shared history conditioning, explicit field transport, and reaction-diffusion refinement. By transporting observed fields once per prediction step and using subsequent layers for refinement, SIDDY reuses observed content while learning the remaining evolution. Across all nine forecasting settings, it achieves the lowest L_2 error with fewer parameters than the baselines. In the future, we plan to extend SIDDY to 3D and point cloud datasets to enhance its applicability in more real-world scenarios.

AI USE STATEMENT

Writing assistance. Generative AI tools were used for language polishing, LaTeX formatting, data organization, and assistance with supplementary descriptions in the appendix. The main manuscript text was written by the authors, who reviewed and revised the AI-assisted writing.

Literature search. Generative AI tools were also used to search for potentially relevant prior work and to locate references on physical concepts, including semi-Lagrangian sampling. The authors are responsible for checking the selected references against their original sources and take responsibility for the final content of the paper.

ETHICS STATEMENT

The experiments use simulated fluid fields, ink videos, and ERA5 reanalysis data, with sources documented in Appendix B. These experiments do not involve human participants or personal data.

REPRODUCIBILITY STATEMENT

Core code for SIDDY is included in the **Supplementary Material**. Implementation details and model configurations are given in Appendices A and C, respectively. Dataset and training settings are provided in Appendices B and D, with loss functions and evaluation metrics in Appendix D.1.

REFERENCES

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 619:533–538, 2023. doi: 10.1038/s41586-023-06185-3.

Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Jayesh K. Gupta, Kit Thambiratnam, Alexander T. Archibald, Chun-Chieh Wu, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. A foundation model for the Earth system. *Nature*, 641:1180–1187, 2025. doi: 10.1038/s41586-025-09005-y.

Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical Fourier neural operators: Learning stable dynamics on the sphere. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2806–2823, 2023. URL https://proceedings.mlr.press/v202/bonev23a.html.

Edoardo Calvello, Nikola B. Kovachki, Matthew E. Levine, and Andrew M. Stuart. Continuum attention for neural operators. *Journal of Machine Learning Research*, 26(300):1–52, 2025. URL https://jmlr.org/papers/v26/24-0879.html.

Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6:631–640, 2024. doi: 10.1038/s42256-024-00844-4.

Shuhao Cao. Choose a transformer: Fourier or Galerkin. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d0921d442ee91b896ad95059d13df618-Abstract.html.

Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=By4HsfWAZ.

Yitong Deng, Hong-Xing Yu, Jiajun Wu, and Bo Zhu. Learning vortex dynamics for fluid inference and prediction. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nYWqxUwFc3x.

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Efficient token mixing for transformers via adaptive Fourier neural operators. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=EXHG-A3jlM.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A general neural operator transformer for operator learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12556–12569, 2023. URL https://proceedings.mlr.press/v202/hao23c.html.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pp. 694–711, 2016. doi: 10.1007/978-3-319-46475-6_43. URL https://link.springer.com/chapter/10.1007/978-3-319-46475-6_43.

Jean Kossaifi, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized Fourier neural operator for high-resolution PDEs. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=AWiDlO63bH.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL https://jmlr.org/papers/v24/21-1524.html.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336.

Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11474–11484, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Le_Guen_Disentangling_Physical_Dynamics_From_Unknown_Factors_for_Unsupervised_Video_Prediction_CVPR_2020_paper.html.

Jin Lee, Ziming Liu, Xinling Yu, Yixuan Wang, Haewon Jeong, Murphy Niu, and Zheng Zhang. KANO: Kolmogorov-Arnold neural operator. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/73bf692447f174984f30499ec9b20e04-Abstract-Conference.html.

Zhihao Li, Zhilu Lai, Xiaobo Zhang, and Wei Wang. M2NO: An efficient multi-resolution operator framework for dynamic multi-scale PDE solvers. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 771–782. Association for Computing Machinery, 2026. doi: 10.1145/3770854.3780179. URL https://doi.org/10.1145/3770854.3780179.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for PDE surrogate modeling. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/590daf74f99ee85df3d8c007df9c8187-Abstract-Conference.html.

Zongyi Li, Nikola 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.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023b. URL https://jmlr.org/papers/v24/23-0064.html.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Animashree Anandkumar. Geometry-informed neural operator for large-scale 3D PDEs. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 35836–35854. Curran Associates, Inc., 2023c. doi: 10.52202/075280-1556. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/70518ea42831f02afc3a2828993935ad-Paper-Conference.pdf.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024. doi: 10.1145/3648506.

Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32576–32594, 2024. URL https://proceedings.mlr.press/v235/liu-schiaffini24a.html.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from data. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3208–3216, 2018. URL https://proceedings.mlr.press/v80/long18a.html.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Till Muser, Alexandra Spitzer, Matti Lassas, Maarten V. de Hoop, and Ivan Dokmanić. Flowers: A warp drive for neural PDE solvers. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=TPpUKH8fuQ.

Carlos A. Pereira, Stéphane Gaudreault, Valentin Dallerit, Christopher Subich, Shoyon Panday, Siqi Wei, Sasa Zhang, Siddharth Rout, Eldad Haber, Raymond J. Spiteri, and David Millard. Learning to advect: A neural semi-Lagrangian architecture for weather forecasting, 2026. URL https://arxiv.org/abs/2601.21151.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03409.

Md Ashiqur Rahman, Zachary E. Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=j3oQF9coJd.

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

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of PDEs. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f3c1951b34f7f55ffaecada7fde6bd5a-Abstract-Conference.html.

Andrew Staniforth and Jean Côté. Semi-Lagrangian integration schemes for atmospheric models—a review. *Monthly Weather Review*, 119(9):2206–2223, 1991. doi: 10.1175/1520-0493(1991)119⟨2206:SLISFA⟩2.0.CO;2.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized Fourier neural operators. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2111.13802.

Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023. doi: 10.1016/j.cma.2022.115783.

Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/43e4e6a6f341e00671e123714de019a8-Abstract.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53681–53705, 2024. URL https://proceedings.mlr.press/v235/wu24r.html.

Lanxiang Xing, Haixu Wu, Yuezhou Ma, Jianmin Wang, and Mingsheng Long. HelmFluid: Learning Helmholtz dynamics for interpretable fluid prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54673–54697, 2024. URL https://proceedings.mlr.press/v235/xing24c.html.

Zherui Yang, Haiyang Xin, Tao Du, and Ligang Liu. Simple yet effective: Low-rank spatial attention for neural operators. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=PhSw9Xhw6e.

Niloufar Zakariaei, Siddharth Rout, Eldad Haber, and Moshe Eliasof. Advection augmented convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/3e25d1aff47964c8409fd5c8dc0438d7-Abstract-Conference.html.

Jinpai Zhao, Nishant Panda, Yen Ting Lin, Eirik Valseth, Diane Oyen, and Clint Dawson. HyCOP: Hybrid composition operators for interpretable learning of PDEs, 2026. URL https://arxiv.org/abs/2605.00820.

A Implementation Details

This appendix specifies the computations underlying Section 3. The input tensor has shape $B \times T \times C \times H \times W$, and the model returns one field of shape $B \times C_{\text{out}} \times H \times W$. For autoregressive prediction of the same variables, $C_{\text{out}} = C$. Table 4 lists reference settings, with dataset-specific settings given in Appendices B and C.

Algorithm 1 One prediction step of SIDDY

Require: History U_n, transport window S, candidates K, refinement depth L
1: Encode $c_n = \mathcal{E}_\theta(U_n)$ and lift the selected frames to $e_{n,j}$
2: Predict displacements and scores from $e_{n,j}$, c_n, and lag embeddings
3: Compute $\widetilde{u}_{n+1}$ and lift v_n using Equations 4–5
4: Initialize $h^{(0)} = v_n + W_{\text{in}}[v_n, c_n]$
5: **for** $\ell = 0, \dots, L-1$ **do**
6: Compute reaction and diffusion and update $h^{(\ell+1)}$ by Equation 9
7: **end for**
8: **return** $\mathcal{B}_\theta(\widetilde{u}_{n+1}) + \mathcal{P}_\theta(h^{(L)})$

A.1 Network Blocks

Notation and refinement fusion. The index n denotes forecast time, whereas $\ell = 0, \dots, L-1$ indexes refinement layers. For feature fields, square brackets denote channel concatenation. In Equation 2, the innermost brackets collect observations across time. The symbol $\odot$ denotes elementwise multiplication, with channel vectors broadcast over spatial locations. Omitting the batch dimension, c_n, v_n, $h^{(\ell)}$, and $f^{(\ell)}$ have shape $d \times H \times W$. In Equation 6, the layer-specific fusion map V_ℓ takes the concatenated current state and history context and returns a d-channel increment according to

$$V_\ell \colon \mathbb{R}^{2d \times H \times W} \to \mathbb{R}^{d \times H \times W}, \qquad f^{(\ell)} = h^{(\ell)} + V_\ell[h^{(\ell)}, c_n]. \tag{10}$$

The map W_{in} initializes the refinement state from $[v_n, c_n]$, while V_ℓ fuses the evolving state with the shared context in each layer. The pointwise affine map G_ℓ produces gate logits from $[h^{(\ell)}, c_n]$, and a sigmoid maps them to the d-channel field $\boldsymbol{g}_\ell$. The set $\mathscr{S}$ contains spatial radii and is distinct from the transport-window length S and the context downsampling factor s.

Temporal and spatial mixing. After reordering the input to $B \times (CT) \times H \times W$, two grouped 1×1 convolutions map $T \to p \to p$ within each physical variable, with GELU after each convolution. Their weights are specific to each variable and shared across grid points. Two further pointwise convolutions map $Cp \to 2d \to d$, with GELU between them. The resulting field passes through a residual convolution block and a multiscale spatial mixer. By mixing time before spatial attention, the encoder produces spatial features without constructing a separate attention token for every location and time step.

The residual convolution block first applies GroupNorm, using the largest divisor of the channel count not exceeding eight. A 3×3 convolution expands the normalized input from d to $2d$ channels, after which GELU and a second 3×3 convolution return it to d channels. A learned channel scale then multiplies this increment before it is added to the input.

The multiscale mixer applies parallel depthwise 3×3 convolutions with dilations $r \in \mathscr{S}$ to a normalized feature field, then concatenates and projects their outputs to d channels. A normalized pointwise network of width $2d$ supplies an additional channel increment, which is summed with the spatial contribution before scaling and addition to the input. The function $\mathcal{S}_\ell$ in Equation 6 denotes the resulting scaled increment rather than the full residual output.

Global context. For $s > 1$, downsampling applies a residual block, depthwise 3×3 convolution, GELU, $s \times s$ average pooling, pointwise convolution, GELU, and a residual block. The reduced field has $N_s = \lfloor H/s \rfloor \lfloor W/s \rfloor$ spatial tokens. A depthwise 3×3 positional convolution is added before flattening the grid. Each attention layer uses pre-LayerNorm multihead attention and a pointwise feed-forward network of expansion ratio two, both with residual connections and learned channel scales. A residual block refines the result before bilinear upsampling with `align_corners=False`.

The parallel spectral branch applies a real two-dimensional FFT to z_n at the original resolution. It learns separate complex channel-mixing tensors for the positive and negative frequency bands along the height axis, retaining at most m_H modes in each band and m_W along the width axis. All other output frequencies are zero before the inverse FFT. A learned channel scale multiplies the spectral increment. This spectral increment is concatenated with the local and attention features. Fusion uses a 1×1 convolution, GELU, and a residual block. With the spectral branch disabled, fusion combines the local and attention features.

Frame lifting and output. Each selected frame is lifted by a $C \to d$ pointwise convolution, GELU, and a stack of residual blocks, after which a learned embedding identifies its position in the transport window. These features are used by the displacement predictor. The post-transport lifting network $\mathcal{L}_{\text{post}}$ consists of a $C \to d$ pointwise convolution, GELU, and one residual block. The decoder $\mathcal{P}_\theta$ uses a residual block followed by pointwise convolutions $d \to d \to C_{\text{out}}$, with GELU between them. The map $\mathcal{B}_\theta$ is a separate $C \to C_{\text{out}}$ pointwise convolution applied to the sampled physical field.

A.2 Transport Parameters and Boundaries

Displacements and candidate weights. The predictor concatenates a frame's d-channel lifted features, shared context, and a second d-channel lag embedding. A shared trunk maps these $3d$ channels to $2d$ channels with a 3×3 convolution and GELU, followed by a multiscale mixer and a residual block. Separate pointwise heads output $2K$ displacement components and K scores. Learned frame biases are added to the scores before the softmax in Equation 4.

Given the raw displacement prediction $\boldsymbol{d}_{n,j,k}$, the implemented offset is

$$\boldsymbol{\delta}_{n,j,k} = a_{\text{disp}} \lambda_j \gamma_k \tanh(\boldsymbol{d}_{n,j,k}) + \boldsymbol{\delta}^{\text{vel}}_{n,j}. \tag{11}$$

The scalar a_{disp} sets the displacement scale, while the lag scales λ_j and candidate scales γ_k are trainable and unconstrained. For $K > 1$, the candidate scales are initialized at evenly spaced values between 0.25 and 1, with a single scale of one when $K = 1$. The lag scales are similarly initialized from 1 for the oldest selected frame to 0.5 for the latest when $S > 1$, and to one when the transport window contains only the most recent observation.

The default path predicts displacements without a supplied velocity field, so $\boldsymbol{\delta}^{\text{vel}}_{n,j} = 0$. When velocity channels (v_x, v_y) and domain lengths (L_x, L_y) are supplied, the optional offset is

$$\boldsymbol{\delta}^{\text{vel}}_{n,j}(x) = -(j+1) \left(\frac{2v_{x,n-j}(x)}{L_x}, \frac{2v_{y,n-j}(x)}{L_y} \right). \tag{12}$$

Every candidate from the same frame receives this velocity offset. The learned component in Equation 11 is scaled by a_{disp}, λ_j, and γ_k.

Sampling and boundary treatment. In Equation 4, x and $\boldsymbol{\delta}_{n,j,k}(x)$ use normalized coordinates; in Equation 8, x indexes grid sites, $\boldsymbol{e}_i$ is a one-cell axis shift, and r counts cells. The two-component $\boldsymbol{d}_{n,j,k}$ in Equation 11 is distinct from the feature width d. For periodic transport, the stored $H \times W$ sites have no duplicated endpoints. Write $q = x + \boldsymbol{\delta}_{n,j,k}(x)$, suppressing the indices below. We extend the field with a one-cell circular halo by copying opposite edges and corners. With `align_corners=True`, we convert q to original continuous pixel indices p, wrap them to $\bar{p}$, and normalize the halo-shifted indices to q^+:

$$\begin{aligned} p_x &= \frac{W-1}{2}(q_x + 1), & p_y &= \frac{H-1}{2}(q_y + 1), \\ \bar{p}_x &= p_x - W \lfloor p_x / W \rfloor, & \bar{p}_y &= p_y - H \lfloor p_y / H \rfloor, \\ q^+_x &= \frac{2(\bar{p}_x + 1)}{W+1} - 1, & q^+_y &= \frac{2(\bar{p}_y + 1)}{H+1} - 1. \end{aligned} \tag{13}$$

The wrap periods are W and H, not the $W-1$ and $H-1$ normalization scales, because the stored endpoints are distinct. The extended tensor has shape $(H+2) \times (W+2)$, and $(\bar{p}_x + 1, \bar{p}_y + 1)$ lies strictly within its outer boundary. Bilinear `grid_sample` uses the same alignment convention and border padding, but its padding setting does not determine these periodic samples. Across a seam,

interpolation uses the last and first grid sites as neighbors without clipping coordinate gradients at the outer border. The interpolant is piecewise differentiable, with possible derivative jumps at grid knots. The reported non-periodic configurations use border padding for transport (Table 7). In Equation 8, out-of-grid indices use periodic extension on Navier–Stokes and border extension on the other datasets.

Laplacian and gating. The channel strengths are $\boldsymbol{\nu}_{\ell,r} = a_{\mathrm{diff}} \operatorname{softplus}(\boldsymbol{\beta}_{\ell,r})$, with $a_{\mathrm{diff}} = 0.03$ and $\boldsymbol{\beta}_{\ell,r}$ initialized to -4. These coefficients control diffusion in the learned feature space, while ν denotes physical viscosity in the Navier–Stokes equation. The vectors $\boldsymbol{\beta}_{\ell,r}$ and $\boldsymbol{\alpha}_\ell$ contain d learned entries each, whereas a_{diff} is a scalar configuration value. The gate in Equation 9 is computed as

$$\boldsymbol{g}_\ell = \operatorname{sigmoid}\Big(G_\ell[h^{(\ell)}, c_n]\Big), \tag{14}$$

where G_ℓ is a pointwise affine map. The channel scale $\boldsymbol{\alpha}_\ell$ is learned separately and initialized to 10^{-2}. Parameters are distinct across refinement layers.

A.3 Initialization and Default Configuration

The displacement and score heads are initialized to zero, while a score bias of eight favors the newest selected frame over the others, whose biases are zero. The final output projection is also zero-initialized, and $\mathcal{B}_\theta$ starts as the identity when input and output channels match. With no supplied velocity and $S = 1$, the initial prediction consequently reproduces the latest field up to interpolation roundoff. For $S > 1$, it is a weighted combination at unchanged coordinates, with total weight $e^8/(e^8 + S - 1)$ on the latest frame.

The input fusion of the refinement network and the final pointwise reaction convolutions are zero-initialized. Residual convolution blocks and multiscale mixers also use zero-initialized output projections with small learned scales. In the context encoder, the last variable-fusion projection and the first final-fusion projection are zero-initialized. Frame and lag embeddings have Gaussian initialization with standard deviation 0.02. These choices initialize the forecast close to the transported state before the learned refinement contribution becomes nonzero.

Table 4: Reference configuration. The transport and conditioning windows are separate settings. Dataset-specific settings are given in Appendices B and C.

Component	Setting	Reference
Feature space	Width d	32
History	Conditioning length T, temporal width p	10, $p = T$
Transport	Window S, candidates K	1, 2
Transport	Displacement scale a_{disp}	0.25
Context	Downsampling factor s	4
Context	Attention layers, heads	2, 4
Context	Fourier modes (m_H, m_W)	$(12, 12)$
Context	Spectral branch, dropout	Enabled, 0
Spatial mixing	Radii $\mathscr{S}$	$\{1, 2, 4\}$
Refinement	Layers L	4
Frame lift	Residual blocks	2
Boundaries	Transport, diffusion	Periodic, periodic

A.4 Computational Cost and Parameter Accounting

Let $N = HW$, $N_s = \lfloor H/s \rfloor \lfloor W/s \rfloor$, and $R = |\mathscr{S}|$. Per-variable temporal mixing costs $O(NC(Tp + p^2))$ before cross-variable fusion. At fixed kernel sizes, full-resolution convolutions cost $O(Nd^2)$ per block, while one reduced-grid attention layer costs $O(N_s^2 d + N_s d^2)$. The spectral branch costs $O(dN \log N + d^2 m_H m_W)$. Both global branches are evaluated once per prediction step, independently of refinement depth L.

Transport predicts its K candidates from a shared per-frame trunk. Sampling observations costs $O(SKNC)$ and is followed by the post-transport lifting network. One Laplacian branch costs

$O(RNd)$ and has Rd trainable coefficients, with fixed spatial stencil entries. The remaining channel-mixing and spatial-mixing operations contribute $O(Nd^2)$ per refinement layer for fixed R. Refinement computation and parameters therefore grow linearly with L.

SIDDY's spectral weights are complex-valued. Counting each real or complex tensor element once, the two $d \times d \times m_H \times m_W$ tensors contain $2d^2 m_H m_W$ parameters, plus the d spectral channel scales. Its total parameter count is

$$P_{\text{elem}} = \sum_{w \text{ real}} \text{numel}(w) + \sum_{w \text{ complex}} \text{numel}(w). \tag{15}$$

Fixed Laplacian kernels are registered as buffers, so the count includes their learned strengths but not the stencil entries themselves.

Measured inference efficiency. Table 5 complements the complexity analysis with FLOPs and inference latency measured on Navier-Stokes at batch size one. Parameter counts use the same model configurations as Table 1.

Table 5: **Inference efficiency on Navier-Stokes at batch size one.**

Method	#P (M)	FLOPs (G)	Latency (s)
FNO	<u>1.189</u>	**0.071**	**0.159**
ADRNet	9.436	50.493	<u>0.164</u>
HelmFluid	9.929	11.909	0.290
M2NO	17.338	34.652	0.172
Flowers	17.330	8.543	0.240
LRSA	14.546	42.734	0.305
SIDDY	**0.968**	<u>4.442</u>	0.303

SIDDY has the fewest parameters at 0.968M and the second-lowest FLOPs at 4.442G. Its FLOPs are 48.0% to 91.2% lower than those of the five baselines other than FNO, which uses 0.071G. SIDDY's latency is 0.303 s, compared with 0.305 s for LRSA, and is higher than those of the remaining baselines. FNO has the shortest latency at 0.159 s.

B Dataset Details

This appendix describes the data sources and forecasting tasks used in Section 4. Table 6 provides a brief summary of the data partitions and forecasting windows used in our experiments.

Table 6: **Dataset summary.** The partition column lists the training, validation, and test sequence counts for the numerical benchmarks. For videos, input length denotes the entire observed prefix. Per-call history lengths are given in Table 8.

Dataset	Resolution	(Input, prediction) length	(train, validation, test) Data partition	Observed state
Navier–Stokes, $\nu = 10^{-3}$	64×64	$(10, 40)$	$(1000, \text{-}, 200)$	Vorticity
Navier–Stokes, $\nu = 10^{-4}$	64×64	$(10, 20)$	$(1000, \text{-}, 200)$	Vorticity
Navier–Stokes, $\nu = 10^{-5}$	64×64	$(10, 10)$	$(1000, \text{-}, 200)$	Vorticity
ERA5-Z500	72×144	$(2, 10)$	$(20425, 2087, 4174)$	Geopotential
Spread Ink, Real 1	256×256	$(100, 50)$	One video sequence	RGB image
Spread Ink, Real 2	256×256	$(126, 63)$	One video sequence	RGB image
Spread Ink, Real 3	256×256	$(93, 46)$	One video sequence	RGB image
Spread Ink, Synthetic 1	256×256	$(100, 200)$	One video sequence	RGB image
Spread Ink, Synthetic 2	256×256	$(67, 133)$	One video sequence	RGB image

B.1 NAVIER-STOKES

The Navier-Stokes benchmark models a two-dimensional incompressible viscous fluid on the periodic unit square (Li et al., 2021). The observed scalar field is vorticity ω, whose evolution satisfies

$$\begin{aligned} \partial_t \omega(\boldsymbol{x}, t) + \boldsymbol{v}(\boldsymbol{x}, t) \cdot \nabla \omega(\boldsymbol{x}, t) &= \nu \nabla^2 \omega(\boldsymbol{x}, t) + f(\boldsymbol{x}), \\ \nabla \cdot \boldsymbol{v}(\boldsymbol{x}, t) = 0, \qquad \omega(\boldsymbol{x}, 0) &= \omega_0(\boldsymbol{x}). \end{aligned} \tag{16}$$

Here $\boldsymbol{v}$ is the velocity, ν is the kinematic viscosity, and f is the vorticity forcing. Initial vorticity fields are sampled from a Gaussian random field, and trajectories are generated with a pseudospectral solver. The source data are computed on a 256×256 grid and downsampled to 64×64, with one recorded frame per unit of physical time (Li et al., 2021).

The three tasks use $\nu = 10^{-3}$, 10^{-4}, and 10^{-5}. Starting from ten observed frames, the model predicts the next 40, 20, and 10 frames, respectively. These correspond to sequences of 50, 30, and 20 frames.

B.2 ERA5-Z500

ERA5-Z500 contains global geopotential fields at the 500 hPa pressure level and uses a 2.5° latitude-longitude grid with 72×144 locations. Its chronological partition assigns 2013-2019 to training, 2020 to validation, and 2021-2022 to testing, with 20,425, 2,087, and 4,174 sequences, respectively (Xing et al., 2024). The forecasting task uses two input fields to predict the next ten fields.

B.3 REAL SPREAD INK VIDEOS

The three real sequences are RGB recordings of fluid motion originally released by Deng et al. (2023); we use the data distributed with HelmFluid (Xing et al., 2024). Each frame has resolution 256×256, and the prediction target is the three-channel image rather than a measured velocity or vorticity field. The videos contain 150, 189, and 139 frames. Chronological observed/test splits give 100/50 frames for Real 1, 126/63 for Real 2, and 93/46 for Real 3 (Xing et al., 2024).

Each video defines a separate temporal extrapolation problem. Samples are formed within the observed prefix, and the test task begins at its end. Table 6 lists the observed-prefix length, while Table 8 gives the per-step history length. At the forecast boundary, SIDDY receives a history window and subsequently updates that window with its own predictions, following the autoregressive procedure in Section 3.

B.4 SYNTHETIC SPREAD INK VIDEOS

The two synthetic sequences originate from the vortex-dynamics dataset of Deng et al. (2023); we use the versions distributed with HelmFluid (Xing et al., 2024). They visualize the advection of a color field by simulated vortical motion on a 256×256 grid. The source simulations use a discrete vortex method, a back-and-forth error compensation and correction (BFECC) advection scheme, and third-order Runge–Kutta integration. The first sequence uses a smooth color background and a first-order Gaussian mollifying kernel. The second uses sharply separated color tiles and a second-order Gaussian kernel, producing a different combination of image structure and flow dynamics (Deng et al., 2023).

Both synthetic videos use a $1 : 2$ observed-to-predicted split, rounded to whole frames: 100/200 frames for Synthetic 1 and 67/133 for Synthetic 2. For Synthetic 2, we use the full 200-frame sequence rather than the 180 frames described by Deng et al. (2023). Each sequence is treated as an individual temporal extrapolation task with RGB prediction targets.

C BASELINE CONFIGURATIONS

Table 7 summarizes the architecture configurations of the baselines and SIDDY across the four dataset groups. The Navier–Stokes column covers all three viscosity settings, while the real and synthetic video columns cover the three real and two synthetic Spread Ink sequences, respectively.

Flowers, HelmFluid, and LRSA use the default configuration files released by their authors. For SIDDY, the width varies by dataset group, and the spectral branch is enabled only on Navier–Stokes. Its network components are described in Appendix A.

Table 7: **Model configurations by dataset group.** Modes are listed as height followed by width. M2NO grid levels are listed in configuration order.

Model	Setting	Navier–Stokes	ERA5-Z500	Real videos	Synthetic videos
FNO	Blocks	4	4	4	4
	Modes (m_H, m_W)	$(12, 12)$	$(18, 36)$	$(24, 24)$	$(24, 24)$
	Width	32	32	64	64
ADRNet	Blocks	3	3	3	3
	Width	128	192	64	64
Flowers	Configuration	Authors' default configuration files			
HelmFluid	Configuration	Authors' default configuration files			
LRSA	Configuration	Authors' default configuration files			
M2NO	Blocks	5	5	5	5
	(k, c)	$(8, 2)$	$(4, 4)$	$(4, 4)$	$(4, 4)$
	Grid levels	$(1, 1, 2, 2, 2, 3)$	$(1, 1, 1)$	$(3, 3, 2, 2, 1)$	$(3, 3, 2, 2, 1)$
SIDDY	Width d	32	64	48	48
	Spectral branch	Enabled	Disabled	Disabled	Disabled
	Boundary padding	Periodic	Border	Border	Border

D Training Settings

Table 8 summarizes the training protocols by dataset and model group. All models output one frame per forward pass. Full-rollout training recursively appends predictions to the input history and differentiates the loss through the output window. Single-step teacher forcing supervises one prediction from a ground-truth history.

Table 8: **Training settings by dataset and model group.** Spread Ink includes all real and synthetic videos. The output window T_{tr} is the number of supervised prediction steps per training example. Learning rates are initial rates for StepLR and peak rates (`max_lr`) for OneCycleLR.

Dataset	ERA5-Z500	Spread Ink		Navier–Stokes	
Models	All	LRSA, M2NO	FNO, ADRNet, HelmFluid, Flowers, SIDDY	LRSA, M2NO, SIDDY	FNO, ADRNet, HelmFluid, Flowers
Input frames	2	10	10	10	10
Output window	10	1	10	1	T_{tr}
Output per call	1	1	1	1	1
Training	Full rollout	Single-step teacher forcing	Full rollout	Single-step teacher forcing	Full rollout
Batch size	5	2	2	Adjusted	Adjusted
Optimizer	Adam	AdamW	Adam	AdamW	Adam
Scheduler	StepLR	OneCycleLR	StepLR	OneCycleLR	StepLR
Learning rate	5×10^{-4}	10^{-3}	10^{-4}	10^{-3}	5×10^{-4}

ERA5-Z500. All models use full-rollout training with Adam and StepLR in the HelmFluid training framework (Xing et al., 2024), with an initial learning rate of 5×10^{-4}. Training runs for 100,000 minibatch iterations. Input lengths, batch sizes, and training lengths are listed in Table 8. Other hyperparameters retain the values in the corresponding training scripts.

Spread Ink. All models are trained for 501 epochs. Across all five videos, LRSA and M2NO use single-step teacher forcing with AdamW and OneCycleLR, with $\texttt{max_lr} = 10^{-3}$. FNO, ADRNet, HelmFluid, Flowers, and SIDDY use ten-step full-rollout training with Adam, StepLR, and an initial learning rate of 10^{-4}. Every model is evaluated autoregressively.

Navier-Stokes. All models are trained for 501 epochs. LRSA and M2NO follow the single-step teacher-forcing procedures in their authors' implementations (Yang et al., 2026; Li et al., 2026). SIDDY uses the same training regime. These three models use ten input frames and a one-frame output window, with AdamW and OneCycleLR at $\texttt{max_lr} = 10^{-3}$. FNO, ADRNet, HelmFluid, and Flowers use full-rollout training with Adam, StepLR, and an initial learning rate of 5×10^{-4}. Their output window is $T_{\mathrm{tr}} = 40$, 20, or 10 for $\nu = 10^{-3}$, 10^{-4}, or 10^{-5}, respectively. Other hyperparameters, including weight decay, retain their implementation defaults. Batch size is adjusted to the available GPU memory.

Evaluation horizons. All models are evaluated autoregressively without teacher forcing. Training rollout lengths are listed in Table 8. The test horizons are 40, 20, and 10 frames for the three Navier-Stokes viscosities, 50, 63, and 46 frames for the real videos, 200 and 133 frames for the synthetic videos, and ten frames for ERA5-Z500, as summarized in Appendix B.

D.1 Training Loss and Evaluation Metrics

Let $u_{b,t}$ and $\widehat{u}_{b,t}$ denote the target and prediction for sample b at forecast step t. Norms flatten the spatial and channel entries on the corresponding evaluation domain.

Training loss. For a minibatch of B samples and a training rollout of T_{tr} steps, the rollout training scripts accumulate the stepwise relative L_2 loss,

$$\mathcal{L}_{\mathrm{train}} = \sum_{b=1}^{B} \sum_{t=1}^{T_{\mathrm{tr}}} \frac{\|\mathcal{Q}_b(\widehat{u}_{b,t}) - \mathcal{Q}_b(u_{b,t})\|_2}{\|\mathcal{Q}_b(u_{b,t})\|_2}. \tag{17}$$

Here $\mathcal{Q}_b$ is the identity for Navier–Stokes, the RGB-to-YUV map $\mathcal{Y}$ for Spread Ink, and the input-history normalization $\mathcal{N}_b$ for ERA5-Z500:

$$\mathcal{Y}(u) = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} u, \qquad \mathcal{N}_b(u) = \frac{u - \mu_b}{v_b}. \tag{18}$$

The color transform is applied at each pixel. For ERA5-Z500, μ_b and v_b are the scalar mean and variance of the sample's input history, shared across all forecast steps.

Test relative L_2 and MSE. Let N_{te} be the number of evaluated samples and T_{te} the prediction horizon. For relative L_2, $z_{b,t}$ denotes the vorticity field for Navier–Stokes, the RGB field for Spread Ink, or $\mathcal{N}_b(u_{b,t})$ for ERA5-Z500; $\widehat{z}_{b,t}$ uses the same representation. The reported error is

$$E_{L_2} = \frac{1}{N_{\mathrm{te}}} \sum_{b=1}^{N_{\mathrm{te}}} \frac{\sqrt{\sum_{t=1}^{T_{\mathrm{te}}} \|\widehat{z}_{b,t} - z_{b,t}\|_2^2}}{\sqrt{\sum_{t=1}^{T_{\mathrm{te}}} \|z_{b,t}\|_2^2}}. \tag{19}$$

MSE uses RGB values for Spread Ink and ERA5-Z500 values after reversing $\mathcal{N}_b$. With D scalar entries per evaluated field, it is

$$E_{\mathrm{MSE}} = \frac{1}{N_{\mathrm{te}} T_{\mathrm{te}} D} \sum_{b=1}^{N_{\mathrm{te}}} \sum_{t=1}^{T_{\mathrm{te}}} \|\widehat{u}_{b,t} - u_{b,t}\|_2^2. \tag{20}$$

VGG perceptual loss. For Spread Ink, we use HelmFluid's implementation (Xing et al., 2024) of the VGG feature loss (Johnson et al., 2016). Let ϕ denote frozen VGG16 features at `relu2_2`, including RGB normalization with channel means $(0.485, 0.456, 0.406)$ and standard deviations $(0.229, 0.224, 0.225)$. If D_ϕ is the number of feature entries per frame, the evaluation metric is

$$E_{\mathrm{VGG}} = \frac{1}{N_{\mathrm{te}} T_{\mathrm{te}} D_\phi} \sum_{b=1}^{N_{\mathrm{te}}} \sum_{t=1}^{T_{\mathrm{te}}} \|\phi(\widehat{u}_{b,t}^{\mathrm{RGB}}) - \phi(u_{b,t}^{\mathrm{RGB}})\|_2^2. \tag{21}$$

VGG loss is used for evaluation only.

E Additional Ablation Studies

E.1 Laplacian Refinement

The fixed Laplacian stencils provide the refinement layers with spatial differences at several radii, and the positive strengths $\boldsymbol{\nu}_{\ell,r}$ weight these responses separately for each feature channel. When the Laplacian response enters the combined update in Equation 9, it is further scaled by the sigmoid gate $\boldsymbol{g}_\ell$ and the learned channel scale $\boldsymbol{\alpha}_\ell$, giving the effective coefficient

$$\boldsymbol{\kappa}_{\ell,r}(x) = \boldsymbol{\alpha}_\ell \odot \boldsymbol{g}_\ell(h^{(\ell)}, c_n)(x) \odot \boldsymbol{\nu}_{\ell,r}. \tag{22}$$

Since $\boldsymbol{\alpha}_\ell$ is learned over $\mathbb{R}^d$, these effective coefficients can take either sign, with their magnitudes also varying across locations through the gate. In this way, the refinement combines a fixed spatial difference pattern with learned control over its contribution to the feature update.

To examine the contribution of this structure, we compare SIDDY with a variant that removes the Laplacian branch from the refinement layers on Navier–Stokes and the two synthetic Spread Ink videos. Table 9 presents the new ablation results alongside the full-model results from Tables 1 and 2. Including the branch reduces relative L_2 error by 15.0%, 2.4%, and 1.4% at $\nu = 10^{-3}, 10^{-4}$, and 10^{-5}, respectively, and by 3.5% and 17.9% on Synthetic 1 and Synthetic 2. On both synthetic videos, the lower L_2 error is accompanied by lower MSE and higher VGG loss.

These gains in field-level accuracy come with a small increase in model size, as the stencil entries are fixed and only their channel-wise strengths are learned. Across L refinement layers, the branch introduces $Ld|\mathscr{S}|$ trainable parameters. With $L = 4$ and $\mathscr{S} = \{1, 2, 4\}$, this amounts to 384 parameters for Navier–Stokes ($d = 32$) and 576 for the synthetic videos ($d = 48$), increasing the respective model sizes by approximately 0.04%. The ablation therefore shows that the fixed spatial structure improves field prediction with very little additional parameterization.

F Physical Diagnostics on Navier–Stokes

We evaluate kinetic energy and enstrophy spectra on the 200 test trajectories at 64×64 resolution. The prediction horizons are 40, 20, and 10 steps for $\nu = 10^{-3}$, 10^{-4}, and 10^{-5}, respectively. All quantities are computed from predicted and reference vorticity in the original physical scale.

Spectral quantities. For each sample and forecast step, let $a_{\boldsymbol{k}}$ be the normalized discrete Fourier coefficient of vorticity on the periodic unit square. The modal enstrophy $z_{\boldsymbol{k}}$ and kinetic energy $e_{\boldsymbol{k}}$ are

$$\begin{aligned} a_{\boldsymbol{k}} &= \frac{1}{HW} \mathcal{F}[\omega]_{\boldsymbol{k}}, & z_{\boldsymbol{k}} &= \frac{1}{2}|a_{\boldsymbol{k}}|^2, \\ e_{\boldsymbol{k}} &= \begin{cases} z_{\boldsymbol{k}}/(4\pi^2\|\boldsymbol{k}\|_2^2), & \boldsymbol{k} \neq \boldsymbol{0}, \\ 0, & \boldsymbol{k} = \boldsymbol{0}. \end{cases} \end{aligned} \tag{23}$$

Here $\boldsymbol{k}$ is the integer wavevector. The kinetic energy definition assumes zero mean velocity. Enstrophy retains the vorticity zero mode. We use the full Fourier transform and sum modes within radial shells to obtain

$$\mathcal{K}_r = \{\boldsymbol{k} : r - \tfrac{1}{2} \le \|\boldsymbol{k}\|_2 < r + \tfrac{1}{2}\}, \qquad E(r) = \sum_{\boldsymbol{k} \in \mathcal{K}_r} e_{\boldsymbol{k}}, \qquad Z(r) = \sum_{\boldsymbol{k} \in \mathcal{K}_r} z_{\boldsymbol{k}}. \tag{24}$$

All discrete modes are retained, including Nyquist modes and the corner modes in shells $r = 32, \ldots, 45$. The totals are $Q_E = \sum_r E(r)$ and $Q_Z = \sum_r Z(r) = \frac{1}{2}\langle \omega^2 \rangle$, where brackets denote the spatial mean.

Table 9: **Laplacian ablation on Navier–Stokes and synthetic Spread Ink videos.** Parameter counts (#P) are exact integers; Δ compares SIDDY with w/o Laplacian.

(a) Navier–Stokes

Setting	#P	L_2		
		$\nu = 10^{-3}$	$\nu = 10^{-4}$	$\nu = 10^{-5}$
w/o Laplacian	**967,465**	0.0040	0.0247	0.0362
SIDDY	967,849	**0.0034**	**0.0241**	**0.0357**
Δ	$\uparrow$0.04%	$\downarrow$15.0%	$\downarrow$2.4%	$\downarrow$1.4%

(b) Synthetic Spread Ink

Setting	#P	Synthetic 1			Synthetic 2		
		L_2	MSE	VGG	L_2	MSE	VGG
w/o Laplacian	**1,501,785**	0.3700	0.0424	**10.2243**	0.3520	0.0843	**9.6110**
SIDDY	1,502,361	**0.3569**	**0.0394**	11.0395	**0.2891**	**0.0569**	11.8296
Δ	$\uparrow$0.04%	$\downarrow$3.5%	$\downarrow$7.1%	$\uparrow$8.0%	$\downarrow$17.9%	$\downarrow$32.5%	$\uparrow$23.1%

Error aggregation. For $S \in \{E, Z\}$, let $S_{b,t}$ and $\widehat{S}_{b,t}$ denote reference and predicted shell spectra. Using the sample and horizon notation from Appendix D.1, we compute

$$\epsilon_S = \frac{1}{N_{\text{te}} T_{\text{te}}} \sum_{b=1}^{N_{\text{te}}} \sum_{t=1}^{T_{\text{te}}} \frac{\|\widehat{S}_{b,t} - S_{b,t}\|_2}{\|S_{b,t}\|_2}, \qquad \eta_S = \frac{1}{N_{\text{te}} T_{\text{te}}} \sum_{b=1}^{N_{\text{te}}} \sum_{t=1}^{T_{\text{te}}} \frac{|\widehat{Q}_{S,b,t} - Q_{S,b,t}|}{Q_{S,b,t}}. \tag{25}$$

Errors are computed for each sample and forecast step before averaging. The spectrum error at the final step averages $t = T_{\text{te}}$ over samples. Table 10 reports these errors as percentages.

Table 10: **Physical diagnostics of SIDDY on Navier–Stokes.** Spectrum errors use relative L_2, and errors in the total quantities use relative absolute differences. Mean errors average all test samples and forecast steps. Final errors average the last step over samples. All errors are percentages.

ν	Steps	E spectrum		Z spectrum		Totals	
		Mean	Final	Mean	Final	η_E	η_Z
10^{-3}	40	0.0488	0.0533	0.0626	0.0799	0.0475	0.0528
10^{-4}	20	0.1330	0.2809	0.1958	0.5499	0.1255	0.1241
10^{-5}	10	0.1394	0.3230	0.1945	0.4628	0.1266	0.1147

Results. Mean kinetic energy spectrum errors range from 0.0488% to 0.1394%, and mean enstrophy spectrum errors from 0.0626% to 0.1958%. Mean errors in the total quantities remain below 0.13% in all three settings. Figure 4 shows the averaged spectra, while Figure 5 shows how spectrum errors averaged over samples evolve. At $\nu = 10^{-4}$ and 10^{-5}, both errors increase over the rollout. At $\nu = 10^{-3}$, the kinetic energy spectrum error peaks before the final step.

The spectra closely match where most energy and enstrophy are concentrated. At $\nu = 10^{-3}$, predicted tails at high wavenumbers exceed the much smaller reference tails. These diagnostics describe the distribution of kinetic energy and enstrophy across scales and complement the L_2 metric on the predicted fields.

G Long-Horizon Rollout on Synthetic 1

Synthetic 1 uses the first 100 video frames as the observed segment and the following 200 frames for autoregressive evaluation (Appendix B.4). Figure 6 plots RGB L_2 error (`l2_rgb`) over these

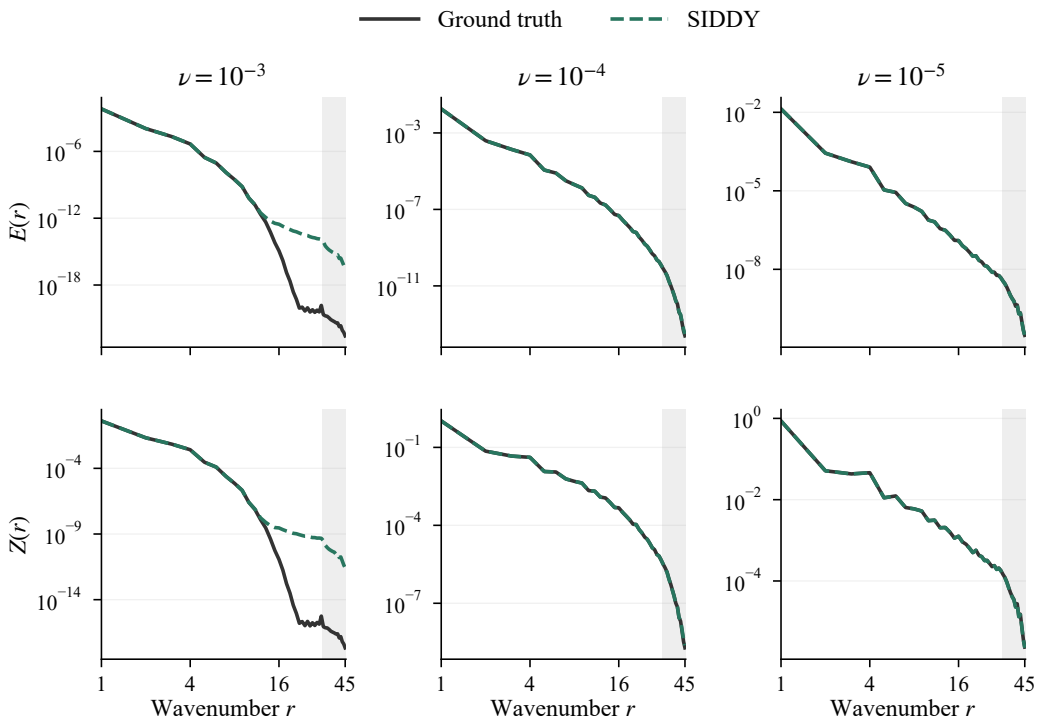

Figure 4: **Kinetic energy and enstrophy spectra on Navier–Stokes.** Curves average the shell sums over all test samples and forecast steps. The top and bottom rows show $E(r)$ and $Z(r)$, respectively. Shading marks shells that intersect the boundary of the square Fourier grid. The logarithmic plots start at $r = 1$.

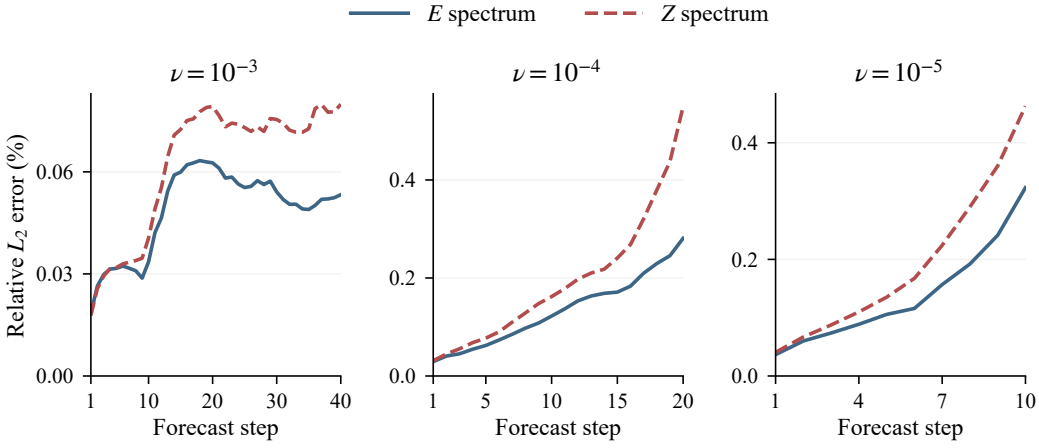

Figure 5: **Spectrum error over the Navier–Stokes rollout.** Each curve averages relative L_2 spectrum errors over samples at the indicated forecast step. Step 1 is the first prediction after the ten input frames.

200 prediction steps. Step 1 is the first prediction after the observed segment. All models use their own predictions at subsequent steps, as described in Appendix D.

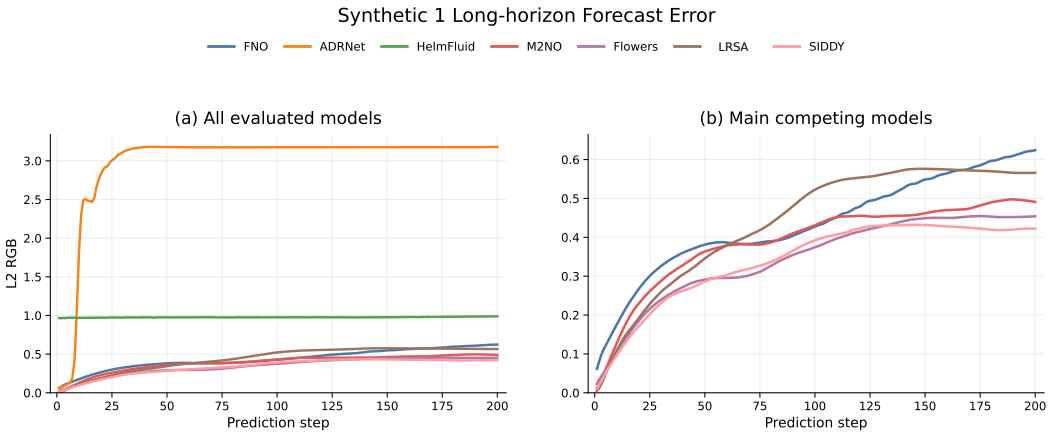

Figure 6: **Full-rollout error on Spread Ink Synthetic 1.** Both panels show RGB L_2 error over all 200 prediction steps. The left panel includes all models. The right panel omits ADRNet and HelmFluid and uses a smaller vertical range to distinguish the other methods.

Lead-time-dependent comparison. Flowers achieves lower error during part of the early and intermediate forecast horizon. SIDDY achieves the lowest error later in the rollout and remains below Flowers through the final step. Over the complete horizon, SIDDY obtains an L_2 error of 0.3569 compared with 0.3636 for Flowers (Table 2).

Field and perceptual errors. RGB L_2 and VGG loss measure field accuracy and differences in pretrained image features, respectively. SIDDY achieves the lowest L_2 error on Synthetic 1, while Flowers achieves the lowest VGG loss (Table 2). The visualizations at selected prediction steps in Appendix H complement the error curves over the full horizon.

H Additional Qualitative Results

Figures 7–11 show additional forecasts for Spread Ink Real 2, Real 3, Synthetic 1, Synthetic 2, and ERA5-Z500. As in Figure 3, each column shows ground truth or the prediction of the method named above it. Each video comparison shows two prediction steps, counted from the start of the test segment. Tables 1 and 2 report results for all baselines. For ERA5-Z500, the columns form two groups of four, with the same two forecast frames in each group.

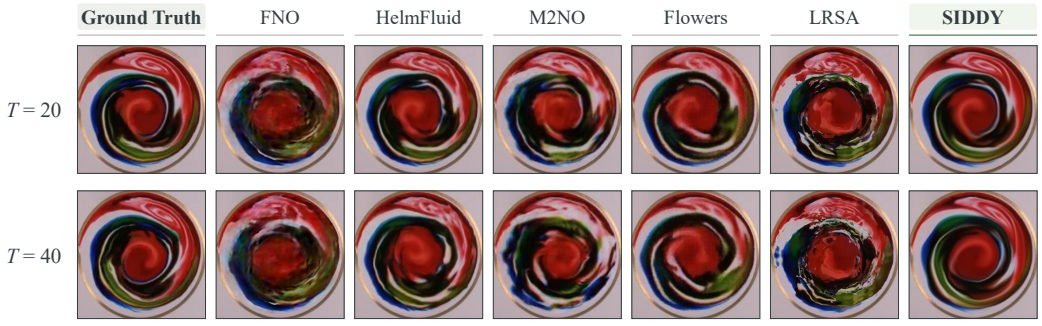

Figure 7: **Prediction results on Spread Ink Real 2.** Columns show ground truth and predictions from the indicated methods at prediction steps 20 and 40.

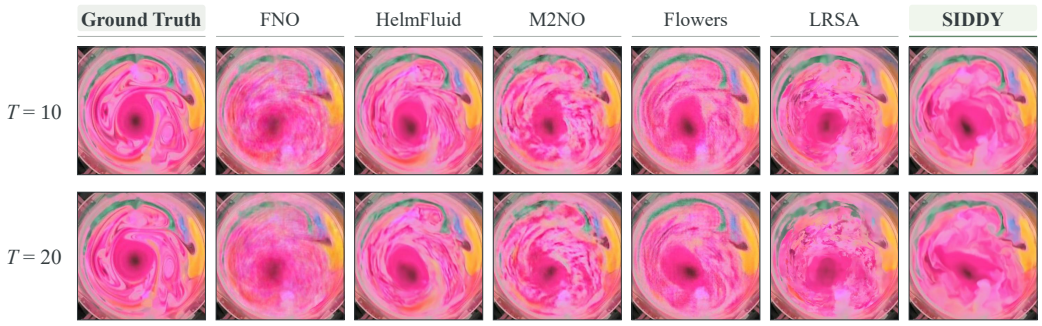

Figure 8: **Prediction results on Spread Ink Real 3.** Columns show ground truth and predictions from the indicated methods at prediction steps 10 and 20.

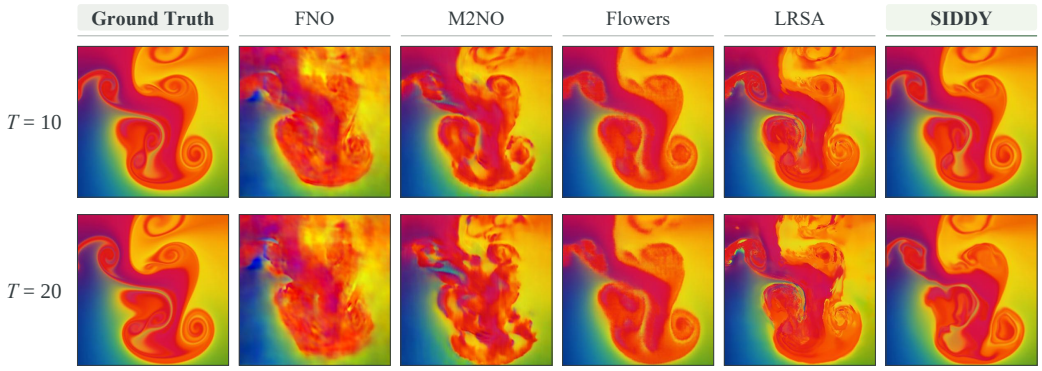

Figure 9: **Prediction results on Spread Ink Synthetic 1.** Columns show ground truth and predictions from the indicated methods at prediction steps 10 and 20.

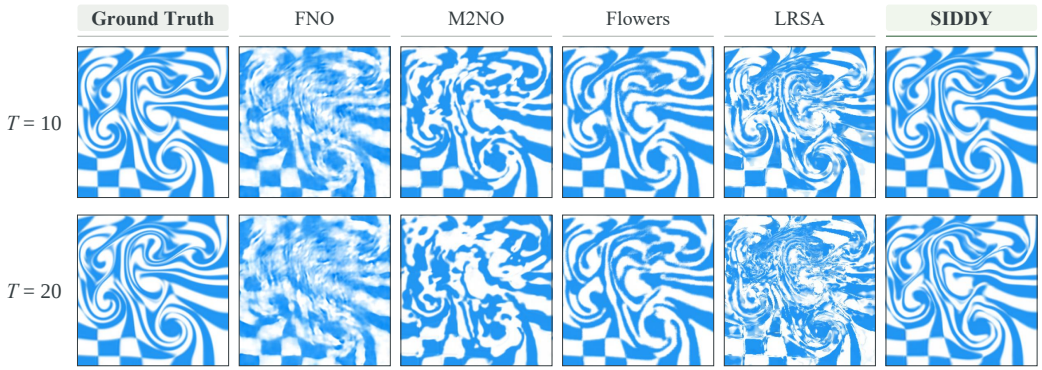

Figure 10: **Prediction results on Spread Ink Synthetic 2.** Columns show ground truth and predictions from the indicated methods at prediction steps 10 and 20.

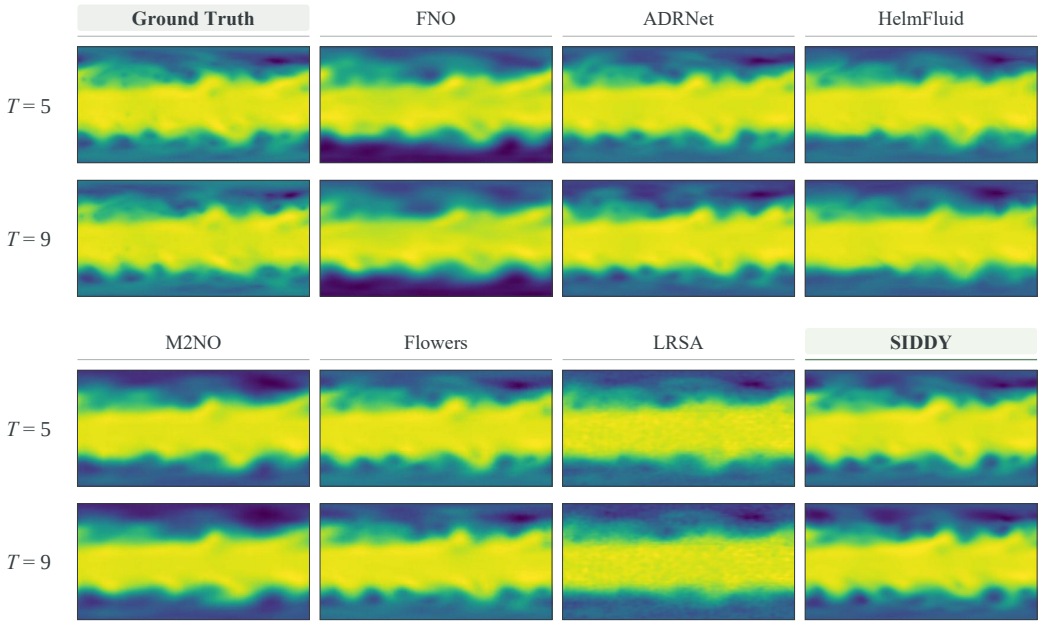

Figure 11: **Prediction results on ERA5-Z500.** Columns show ground truth and predictions from the indicated methods at frames 5 and 9. The columns are arranged in two groups of four, with the same two forecast frames shown in each group.

I Limitations and Future Work

Long-horizon accuracy. The Synthetic 1 curves in Appendix G show error accumulation and changes in relative performance over the forecast horizon. Improving long-horizon accuracy and preserving visual detail remain important directions for future work.

Three-dimensional flows. The current architecture operates on two-dimensional fields. A three-dimensional extension must address the memory required for historical features and sampled candidates, together with the cost of global context. FactFormer (Li et al., 2023a) uses axial kernel factorization to reduce the cost of multidimensional attention. MG-TFNO (Kossaifi et al., 2024) combines low-rank spectral parameterization with multigrid domain decomposition to reduce parameter and memory requirements. One direction is to adapt these techniques by sharing historical context across smaller spatial subdomains. Such a design would use coarse global information and neighboring samples to guide transport and refinement within each subdomain, reducing the fine-resolution data processed at once while retaining interactions across the domain.

Point clouds and unstructured meshes. SIDDY uses regular grids for interpolation and fixed spatial stencils. Extending it to point clouds and unstructured meshes requires a representation that supports sampling and refinement on irregular domains. GINO (Li et al., 2023c) uses graph-based integral operators to encode irregular inputs on a regular latent grid and decode predictions at query points. Following this approach, one direction is to transfer observed physical values and learned features to an auxiliary grid, perform transport and refinement there, and decode the result at the original query points. In this design, geometry information such as a signed distance function would guide sampling and boundary treatment. This requires jointly designing the grid transfers, transport, and refinement modules.