Adaptive Coordinate Transforms for Neural Operators

Chaoyu Liu[1,*] **Zhonghao Li**[2,*] **Gaohang Chen**[3] **Zakhar Shumaylov**[1]

Zhongying Deng[1,4] **Qian Zhang**[2] **Zhonghua Qiao**[3] **Carola-Bibiane Schönlieb**[1]

[1] Department of Applied Mathematics and Theoretical Physics, University of Cambridge
[2] School of Science, Harbin Institute of Technology, Shenzhen
[3] Department of Applied Mathematics, The Hong Kong Polytechnic University
[4] Current affiliation: Department of Radiology, University of Cambridge

Abstract

Neural operators have achieved promising performance on partial differential equations (PDEs), but most existing models are built on fixed Eulerian coordinates. This mismatch between evolving physical structures and static coordinates creates spatial misalignment, leading to unnecessarily non-local operator mappings and reinforcing a smoothness preference near sharp transitions. Inspired by adaptive coordinate transformations in classical PDE analysis, we propose the Adaptive Coordinate Transform (ACT) block, a plug-and-play module for data-driven geometric adaptation in neural operators. ACT blocks resolve this structural limitation by learning adaptive coordinate systems within the operator learning pipeline. Specifically, given an input feature, the ACT block learns a coordinate transformation and represents the same feature under the transformed coordinates via differentiable sampling. This operation preserves the underlying signal while changing its spatial representation, equivalent to expressing the same physical quantity in different coordinate systems. By adapting the coordinate system to the data, ACT allows the network to better track evolving structures, reduce operator complexity, and dynamically focus on critical features to improve learning. We evaluate the proposed approach across diverse PDE benchmarks and multiple neural operator architectures. Experimental results demonstrate consistent and significant improvements in predictive accuracy, indicating that learning coordinate systems provides a powerful mechanism for enhancing operator learning. The code is publicly available at this repository: https://github.com/Maxwell-996/ACT-NO.

1 Introduction

Neural operators have emerged as a powerful framework for solving partial differential equations (PDEs) [Li et al., 2023a,b, Kovachki et al., 2023, Lu et al., 2021, Bryutkin et al., 2024]. By learning mappings between input and output fields, they can serve as efficient surrogates for classical numerical solvers. To date, the field has been primarily driven by diverse architectural paradigms, enabling modeling of increasingly more complex physical dynamics. Ranging from spectral mixing in the Fourier Neural Operators (FNO) [Li et al., 2021] and locality-aware hierarchical extraction in convolutional encoder-decoders [Liu et al., 2022], to long-range interaction modeling in transformer-based architectures like Transolver [Wu et al., 2024].

Despite their success, a fundamental limitation exists: most existing neural operators are built upon fixed Eulerian coordinates. This static coordinate system is inherently misaligned with the

*Equal contribution. Correspondence to: cl920@cam.ac.uk

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

evolving physical structures (e.g., propagating waves, moving vortices) that characterize many PDEs [Zhang et al., 2024b]. This *spatial misalignment* leads to highly non-local and complex input-output mappings, significantly increasing learning complexity and hindering generalization [Hu et al., 2024]. Furthermore, these operators exhibit a secondary challenge: a *smoothness preference* [Rahaman et al., 2019]. Constrained by their inherent inductive biases, they tend to favor overly smooth mappings and struggle to maintain the correct regularity of the solution, often blurring high-frequency physical phenomena like shocks or sharp transitions.

From the perspective of classical PDE analysis, these challenges of complexity and regularity are primarily consequences of using fixed Eulerian coordinates. Rather than solving such evolving dynamics within a static coordinate system, classical methods often employ adaptive coordinate transformations, ranging from Arbitrary Lagrangian-Eulerian (ALE) formulations [Hirt et al., 1974] to moving mesh methods [Huang and Russell, 2010], to dynamically reshape the computational coordinates. By allowing the coordinate system to track advecting structures and locally compress around sharp gradients, these techniques simultaneously reduce spatial misalignment and avoid excessive numerical smoothing. Consequently, they unravel complex spatiotemporal evolution into a smoother, more localized, and better resolved form.

Inspired by these classical principles of adaptive coordinate transformation, we propose the Adaptive Coordinate Transform (ACT) block, a plug-and-play module designed to equip neural operators with data-driven geometric realignment capabilities. By predicting a coordinate adjustment field from the current feature map, the ACT block resamples intermediate features via differentiable sampling and integrates them through a residual connection. Specifically, we interleave ACT blocks layer-wise within foundational operator backbones (e.g., FNO, CNextU, and Transolver), creating an architecture capable of continuous geometric realignment.

Crucially, we find that ACT blocks act as progressive spatial trackers, realigning evolving features to neutralize spatial misalignment and maintain a stable latent space. By implicitly compressing the local coordinate space, they leverage the Jacobian multiplier (22) to explicitly amplify spatial gradients, effectively counteracting the smoothness preference and recovering sharp physical transitions.

In summary, our work bridges the gap between classical adaptive mesh techniques and modern operator learning. The main contributions are as follows:

- We propose the Adaptive Coordinate Transform (ACT) block, a learnable geometric realignment module inspired by classic PDE techniques, specifically designed to overcome spatial misalignment and the smoothness preference of neural operators built on fixed Eulerian coordinates.
- Extensive experiments across diverse foundational architectures (FNO, CNextU, and Transolver) demonstrate that ACT-augmented models significantly improve predictive accuracy and better capture complex dynamics.
- We reveal an autonomous role decoupling phenomenon in layer-wise networks: intermediate ACTs track advection for latent alignment, while the terminal ACT reconstructs sharp gradients via Jacobian-driven spatial compression.

2 Related Work

Neural Operators on Fixed Coordinates Neural operators learn mappings between function spaces and have achieved strong performance on PDEs. Representative paradigms include spectral operators such as the Fourier Neural Operator (FNO) [Li et al., 2021], encoder-decoder baselines built on modern convolutional blocks [Liu et al., 2022], and transformer-based architectures such as Transolver [Wu et al., 2024]. These models offer distinct inductive biases for spectral mixing, locality-aware hierarchical feature extraction, and long-range interaction modeling, respectively. However, these approaches generally assume a fixed Eulerian coordinate system and do not explicitly model coordinate transformations. For dynamics involving transport and advection, they address spatial misalignment only indirectly through spectral global coupling, stacked local convolutions, or attention-based global interactions, so the operator remains unnecessarily non-local in the original coordinates [Hu et al., 2024, Zhang et al., 2024b].

Coordinate Adaptation in PDE Solvers and PDE Learning Reducing complexity through adaptive coordinate transformations or mesh adaptation is standard in PDE computation. In learning-

based settings, some works learn deformation maps or adaptive meshes to improve discretization and solver efficiency [Rowbottom et al., 2025, Grillo et al., 2025, Zhang et al., 2024a, Hu et al., 2024], while others use learned mappings to parameterize irregular domains or canonicalize data before operator learning [Li et al., 2023a, Shumaylov et al., 2025, Oettinger et al., 2026]. These approaches are related in spirit, but they mainly target mesh optimization, domain parameterization, or preprocessing rather than layer-wise geometric realignment of latent operator representations.

Learnable Deformation Modules Learnable deformation is well established in deep learning [Jaderberg et al., 2015, Detlefsen et al., 2018, Zhao et al., 2025, Freifeld et al., 2017], particularly in image registration [De Vos et al., 2017, Balakrishnan et al., 2019], deformable convolution [Dai et al., 2017], and deformable attention [Jaderberg et al., 2015, Zhu et al., 2021]. These methods provide architectural inspiration for ACT through learned offsets and differentiable sampling. Beyond vision applications, Advection-Augmented CNNs [Zakariaei et al., 2024] and Flowers [Muser et al., 2026] employ learnable deformation modules to model advection and transport in dynamical systems. While these approaches primarily exploit deformation to model physical transport, ACT employs it more generally as a layer-wise feature realignment mechanism within neural operators, reducing spatial misalignment and simplifying the learned operator across different PDEs.

3 Method

This section presents our framework for learning neural operators under adaptive coordinate transformations. We first describe the spatial misalignment induced by fixed Eulerian coordinates, then introduce the proposed ACT block, and finally show how it is integrated with a neural operator backbone.

3.1 Problem Setup

Let $\Omega \subset \mathbb{R}^d$ be a bounded spatial domain equipped with a fixed Eulerian coordinate system. Let $U(\Omega \times [0,T]; \mathbb{R}^{d_u})$ and $V(\Omega \times [0,T]; \mathbb{R}^{d_v})$ denote the input and output function spaces, respectively. We consider an operator $\mathcal{F}$ that maps functions from U to V as:

$$\mathcal{F} : U \to V, \quad u(x,t) \mapsto v(x,t) = \mathcal{F}[u](x,t), \tag{1}$$

which typically corresponds to the solution operator of the underlying PDE. Here $u \in U$ and $v \in V$ are spatiotemporal functions evaluated at a given point $x \in \Omega$ and time $t \in [0,T]$. Depending on the task, $u(x,t)$ may represent the initial and boundary conditions or a forcing term, while $v(x,t)$ typically represents the solution field at a future time step.

Neural operators approximate $\mathcal{F}$ by a parameterized network $\mathcal{F}_\theta$, which is typically formulated as:

$$\mathcal{F}_\theta = \mathcal{Q} \circ \mathcal{G}_\theta^{L-1} \circ \cdots \circ \mathcal{G}_\theta^0 \circ \mathcal{P}, \tag{2}$$

where $\mathcal{P}$ pointwise lifts the input function to a latent feature space, $\mathcal{G}_\theta^l$ denotes the l-th latent operator update (e.g., a spatial integral operator using Fourier or attention mechanisms) acting on the entire domain, and $\mathcal{Q}$ is a local projection mapping the final hidden representation back to the physical output space at x.

3.2 Spatial Misalignment and Geometric Realignment

Physical dynamics are inherently characterized by continuous transport and diffusion, causing structures (e.g., a propagating wave or a moving vortex) to naturally advect and spread across the spatial domain over time. Learning difficulties can therefore arise due to the inherent mismatch between fixed Eulerian grids and continuously evolving physical structures, resulting in a highly non-local and complex input-output mapping.

For the neural operator $\mathcal{F}_\theta : u(x,t) \mapsto v(x,t)$ in Eq. (2), this means that a large part of the apparent complexity may come from spatial misalignment in the representation rather than from the underlying dynamics themselves.

However, classical PDE analysis provides a clear solution to this problem: rather than solving such evolving dynamics within a static coordinate system, classical methods often employ coordinate

transformations to simplify the problem. Concretely, we can define a coordinate transformation

$$\psi : \Omega \to \Omega, \quad x \mapsto \xi := \psi(x), \tag{3}$$

that maps the original Eulerian coordinates to a new coordinate system that inherently tracks the evolving structures. Then the neural operator can be reformulated under the new coordinates as:

$$\bar{\mathcal{F}}_\theta : \bar{u}(\xi, t) \mapsto \bar{v}(\xi, t), \tag{4}$$

where $\bar{u}(\xi, t) = u\left(\phi(\xi), t\right), \bar{v}(\xi, t) = v\left(\phi(\xi), t\right)$ and $\phi := \psi^{-1}$ denotes the inverse map. Therefore, the transformed operator $\bar{\mathcal{F}}_\theta$ may admit a simpler representation, since adaptive coordinate transformations can reduce spatial misalignment and reorganize complex spatiotemporal evolution into a smoother and more localized form [Hirt et al., 1974, Huang and Russell, 2010].

3.3 Adaptive Coordinate Transform (ACT) Block

These observations indicate that geometric variability should be handled explicitly at the level of coordinate representation, rather than being implicitly absorbed by the operator backbone. To this end, we propose the Adaptive Coordinate Transform (ACT) block, which learns feature-conditioned coordinate transformations and applies them to intermediate representations between operator layers.

Let $h^x : \Omega_x \to \mathbb{R}^c$ denote a latent feature function represented in the source coordinate x. The ACT block produces another feature function $\widetilde{h}^\xi : \Omega_\xi \to \mathbb{R}^c$ represented in the target coordinate ξ. In our setting, Ω_x and Ω_ξ correspond to the same physical domain, but they represent the feature under different coordinate parameterizations. Thus, ACT defines a nonlinear operator

$$\mathcal{T}_\theta : h^x \mapsto \widetilde{h}^\xi, \tag{5}$$

which maps one feature function to another while preserving the underlying spatial domain.

The key idea is to learn feature-dependent coordinate maps from the target coordinates to the source coordinates. For each head $m = 1, \ldots, M$, ACT predicts a coordinate offset field

$$\Delta_h^{(m)} : \Omega_\xi \to \mathbb{R}^d, \tag{6}$$

and defines the sampling map

$$\phi_h^{(m)}(\xi) = \iota(\xi) + \Delta_h^{(m)}(\xi), \qquad \xi \in \Omega_\xi, \tag{7}$$

where $\iota : \Omega_\xi \to \Omega_x$ denotes the identity correspondence between the target and source coordinate systems. When $\Delta_h^{(m)} \equiv 0$, the map reduces to identity sampling. Hence, ACT learns an incremental coordinate reparameterization rather than a completely new global coordinate system.

The overall architecture is illustrated in Figure 1. It consists of three operator-level components: feature-conditioned coordinate map prediction, pullback under the learned coordinates, and feature aggregation.

Step 1: Feature-conditioned coordinate map prediction. We first construct head-specific value fields by applying pointwise projections to the source feature:

$$V_m^x(x) = W_v^{(m)} h^x(x), \qquad m = 1, \ldots, M. \tag{8}$$

To predict a coordinate map for each target location ξ, we evaluate the source value field at the identity-corresponding location $\iota(\xi)$ and concatenate it with the target coordinate:

$$f_{\text{in}}^{(m)}(\xi) = \left[V_m^x(\iota(\xi)), \xi\right]. \tag{9}$$

This feature-coordinate field provides both local feature information and explicit positional information for coordinate prediction.

At the continuous level, the raw coordinate offset is predicted by a localized operator

$$\Delta_{h,\text{raw}}^{(m)} = \mathcal{D}_\theta\big(f_{\text{in}}^{(m)}\big), \tag{10}$$

where $\mathcal{D}_\theta$ maps a feature-coordinate field on Ω_ξ to a vector field on Ω_ξ. In our implementation, $\mathcal{D}_\theta$ is realized by a multi-scale convolutional predictor:

$$\Delta_{h,\text{raw}}^{(m)} = \text{Conv}_{\text{fuse}}\Big(\sigma\big([\text{Conv}_{1\times1}(f_{\text{in}}^{(m)}), \text{Conv}_{3\times3}(f_{\text{in}}^{(m)}), \text{Conv}_{\text{dilated}}(f_{\text{in}}^{(m)})]\big)\Big), \tag{11}$$

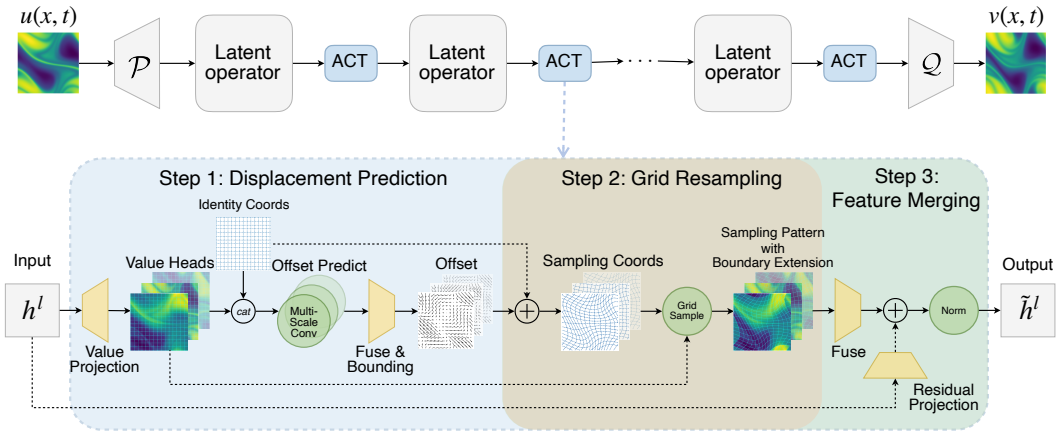

Figure 1: Architecture of the ACT-enhanced operator model.

where σ denotes the GELU activation. This implementation can be viewed as a grid-based discretization of a localized feature-to-vector-field operator.

To stabilize the coordinate transformation, the offset magnitude is bounded by

$$\Delta_h^{(m)} = \alpha \tanh\left(\Delta_{h,\mathrm{raw}}^{(m)}\right), \tag{12}$$

where α controls the maximum coordinate deviation.

Step 2: Pullback under learned coordinates. Given the sampling map $\phi_h^{(m)} : \Omega_\xi \to \Omega_x$, the output value field in target coordinates is defined by

$$\widetilde{V}_m^\xi(\xi) = \left(\mathcal{P}_{\phi_h^{(m)}} V_m^x\right)(\xi) := V_m^x\left(\phi_h^{(m)}(\xi)\right). \tag{13}$$

Here, $\mathcal{P}_{\phi_h^{(m)}}$ denotes the pullback induced by the coordinate map $\phi_h^{(m)}$.

This formulation shows that ACT maps a source feature function to a target feature function by re-indexing where the feature is evaluated. It does not change the domain or output an object outside the function space. On a discrete grid, this pullback is implemented by differentiable interpolation sampling.

To handle out-of-domain coordinates, we impose physically meaningful boundary conditions. Standard differentiable sampling operatorstypically handle such coordinates by clamping them to the nearest border Dai et al. [2017], Zhu et al. [2021]. To better respect the underlying physical topology, ACT instead maps out-of-bounds sampling locations back into the valid domain before grid sampling. For example, in a 1D periodic domain $\Omega = [-1, 1]$, an out-of-bounds coordinate is mapped back into the domain via

$$\hat{\phi}(\xi) = \big((\phi(\xi) + 1) \bmod 2\big) - 1. \tag{14}$$

This coordinate extension ensures that subsequent sampling queries physically consistent locations across periodic boundaries, rather than simply replicating edge values.

Step 3: Feature aggregation. Finally, the transformed value fields are aggregated to produce the output feature function:

$$\widetilde{h}^\xi = \mathcal{T}_\theta(h^x) = \mathrm{Norm}\Big(W_{\mathrm{res}} h^x \circ \iota + W_o\big(\mathrm{Concat}(\widetilde{V}_1^\xi, \ldots, \widetilde{V}_M^\xi)\big)\Big). \tag{15}$$

Here, W_o is a pointwise output projection, and W_{res} is either the identity map or a pointwise projection for channel alignment. The residual term is evaluated on the target coordinate via the identity correspondence ι.

Equivalently, ACT can be written as

$$\mathcal{T}_\theta(h^x) = \mathcal{A}_\theta\left(\mathcal{P}_{\phi_h^{(1)}} W_v^{(1)} h^x, \ldots, \mathcal{P}_{\phi_h^{(M)}} W_v^{(M)} h^x; W_{\mathrm{res}} h^x \circ \iota\right), \tag{16}$$

where $\mathcal{A}_\theta$ denotes the final pointwise aggregation. Since the sampling maps $\phi_h^{(m)}$ are predicted from h^x, ACT is generally nonlinear. Nevertheless, it remains a well-defined operator from feature functions to feature functions.

3.4 Instantiation with Neural Operator Backbone

To prevent spatial misalignment from re-emerging as features propagate through depth, geometric realignment is introduced in a layer-wise manner rather than only once at the input or output. Denoting the operator update at layer l by $\mathcal{G}_\theta^l$ and the corresponding adaptive coordinate block by $\mathcal{T}_\theta^l$, the hidden features evolve as

$$h^{l+1} = \mathcal{T}_\theta^l \circ \mathcal{G}_\theta^l(h^l), \qquad l = 0, \dots, L-1. \tag{17}$$

Accordingly, the overall network takes the form

$$\mathcal{F}_\theta = \mathcal{Q} \circ \mathcal{T}_\theta^{L-1} \circ \mathcal{G}_\theta^{L-1} \circ \cdots \circ \mathcal{T}_\theta^0 \circ \mathcal{G}_\theta^0 \circ \mathcal{P}. \tag{18}$$

In our implementation, each latent operator block is followed by an ACT block, so the hidden stack alternates between operator propagation and geometric realignment. The lifting and projection operators $\mathcal{P}$ and $\mathcal{Q}$ are standard pointwise layers; further details on resampling under learned coordinates and regularization are deferred to Appendix A.

4 Experiments

This section first describes the experimental setting, including the datasets, baselines, evaluation metric, and ACT-specific design choices, and then presents the main evaluation and ablation results.

4.1 Experimental Setting

Dataset Description: We evaluate six time-dependent PDE datasets: `Navier_Stokes`, generated using the FNO data-generation setup [Li et al., 2021]; `Diff_Reaction` and `Shallow_Water` from PDEBench [Takamoto et al., 2022a]; and `Active_Matter`, `TRL_2D`, and the three-dimensional `MHD_64` from The Well [Ohana et al., 2024]. Dataset statistics and descriptions are provided in Table 4 and Appendix B.

Baseline Models: To rigorously evaluate the potential of our proposed ACT mechanism, we integrate it into three representative state-of-the-art baselines spanning distinct paradigms: CNextU [Liu et al., 2022], Transolver++ [Wu et al., 2024, Luo et al., 2025] (an attention-based operator), and FNO [Li et al., 2021] (a classic spectral operator). By applying ACT to these diverse baseline models, we ensure a systematic assessment of its universal effectiveness and modeling enhancements. For a fair comparison, each ACT-enhanced model uses the same backbone configuration, training schedule, optimizer, data split, and evaluation protocol as its corresponding baseline; the only architectural modification is the insertion of ACT blocks. This controlled setup allows us to assess whether adaptive coordinate learning consistently improves different operator architectures.

Metrics: We evaluate model performance using the *Normalized Root Mean Squared Error* (NRMSE), which measures the discrepancy between the predicted field x and the ground-truth field y while being invariant to the scale of the target field. The metric is defined as

$$\mathrm{NRMSE}(x, y) = \sqrt{\frac{\sum_{i=1}^{N} |x_i - y_i|^2}{\sum_{i=1}^{N} |y_i|^2 + \varepsilon}}, \tag{19}$$

where x_i and y_i denote the predicted and ground-truth values at the i-th spatial location, respectively, N is the total number of spatial grid points, and $\varepsilon = 10^{-7}$ is a small constant introduced for numerical stability.

Task Setting: The time-dependent models predict the next state from four consecutive states. At test time, free rollout uses the first four ground-truth states only and feeds each prediction back into the sliding history window:

$$\widehat{u}_{t+1} = \mathcal{F}_\theta(\widetilde{u}_{t-3}, \dots, \widetilde{u}_t). \tag{20}$$

Table 1: **Model Performance Comparison.** NRMSE are reported. Gray columns correspond to models augmented with ACT. Values marked with ↓ indicate the relative error reduction compared to the vanilla (VAN) setting. The best results are highlight in **Bold**.

Dataset	CNextU		Transolver		FNO	
	VAN	+ACT	VAN	+ACT	VAN	+ACT
Navier_Stokes ($\times 10^{-2}$)	6.13	**4.96** ↓19.1%	32.29	**1.35** ↓95.8%	2.22	**0.62** ↓72.1%
Diff_Reaction ($\times 10^{-3}$)	4.03	**1.20** ↓70.2%	1.66	**0.69** ↓58.4%	2.20	**1.34** ↓39.1%
Shallow_Water ($\times 10^{-4}$)	2.11	**1.58** ↓25.1%	3.49	**0.78** ↓77.7%	1.38	**0.93** ↓32.6%
Active_Matter ($\times 10^{-2}$)	3.43	**2.31** ↓32.7%	18.61	**4.91** ↓73.6%	4.49	**2.69** ↓40.1%
TRL_2D ($\times 10^{-2}$)	9.14	**7.57** ↓17.2%	20.34	**6.98** ↓65.7%	11.97	**8.95** ↓25.2%
MHD_64 ($\times 10^{-2}$)	12.08	**9.63** ↓20.3%	29.30	**6.82** ↓76.7%	15.40	**9.09** ↓41.0%
Average ↓	—	**↓30.8%**	—	**↓74.7%**	—	**↓41.7%**

Table 2: Computational cost on Shallow_Water. All models use the same input resolution on an NVIDIA A100 with batch size 32. Inference time is measured per batch. Measurement details are provided in Appendix C.5.

Model	Params (M)	GFLOPs	Memory (MiB)	Inference (s)
CNextU	18.57	9.58	1120	0.038
ACT-CNextU	19.57	13.98	4278	0.072
FNO	16.83	1.70	1410	0.029
ACT-FNO	16.87	4.23	1812	0.054
Transolver	0.18	13.25	3640	0.128
ACT-Transolver	0.33	33.79	4606	0.210

Our Design: Building upon the baseline models, we augment each backbone with ACT blocks that perform data-driven coordinate adaptation. For large backbones, ACT introduces only a small relative parameter overhead: 0.27% for FNO and 5.37% for CNextU. Since Transolver++ is itself a compact architecture, the relative overhead is larger in percentage terms; detailed parameter counts for all models are provided in Appendix C.3. These modules are interleaved with the backbone layers so that intermediate features are repeatedly represented in adaptive coordinate systems, while preserving the original inductive biases of each baseline architecture. For hierarchical architectures such as CNextU, where the feature-map resolution changes across depth, we use a depth-adaptive maximum displacement magnitude(i.e., α in Eq. (12)), that scales with the local grid spacing. Detailed architectural and training configurations are deferred to Appendix C. Finally, to ensure a fair and rigorous evaluation, all other model settings—including network depth, hidden width, training protocols, and optimization hyperparameters—remain strictly identical to those of the corresponding vanilla baselines.

4.2 Evaluation Results

Benchmark Comparison: Across the six datasets in Table 1, ACT reduces one-step NRMSE for all three backbones, with average relative reductions of 74.7%, 41.7%, and 30.8% for Transolver, FNO, and CNextU, respectively. ACT also lowers rollout errors on NS, SW, and MHD_{64}. Appendix E reports gradient, spectral, physical-consistency, and learned-map diagnostics, together with steady elliptic results and comparisons with FLOWERS and ADRNet. Additional visualizations are provided in Appendix F. Table 2 reports computational costs: ACT increases GFLOPs by 1.46–2.55$\times$ and inference latency by 1.64–1.89$\times$. Measurement details are provided in Appendix C.5.

Ablation Study: We conduct a controlled ablation study on the Navier-Stokes equation using FNO as the backbone to disentangle three factors: increased model capacity, a single terminal realignment, and layer-wise realignment throughout the network. The results are summarized in Table 3. Specifically, we compare four variants: (i) the vanilla FNO ($\sim$ 16.8M parameters), (ii) a

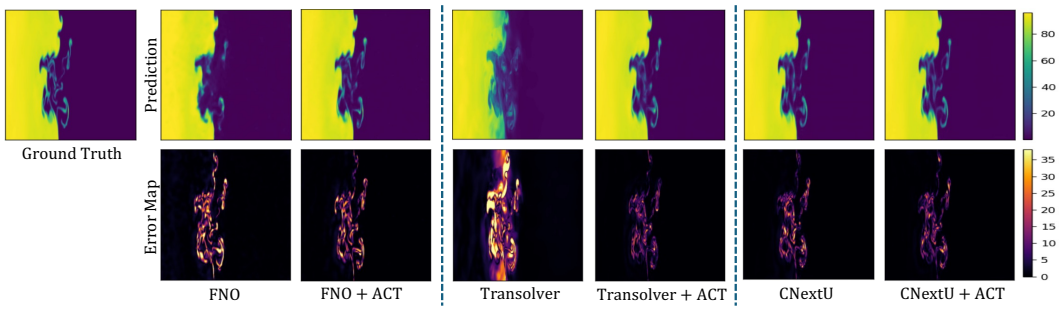

Figure 2: Predicted density ρ results of `TRL_2D` dataset for different models.

Table 3: Ablation study on the Navier–Stokes equation using FNO as the backbone. Introducing adaptive coordinate transformation (ACT) modules significantly improves performance compared to increasing model capacity.

Method	FNO	FNO ($2\times$ params)	FNO + Final ACT	FNO + Layer-wise ACT
NRMSE	2.22%	1.80%	1.15%	**0.62%**

larger FNO with approximately twice the number of parameters ($\sim$ 37.9M), (iii) FNO augmented with only one ACT block at the last layer, and (iv) FNO with ACT blocks applied at every layer. This comparison isolates whether the observed gains arise merely from more parameters or from where ACT is inserted.

The results show that simply increasing model capacity yields only limited improvement. In contrast, introducing only a final ACT block already produces a substantially larger gain. Applying ACT blocks in a layer-wise manner yields a further improvement over both the larger-capacity baseline and the final-ACT-only variant. This controlled comparison indicates that the performance gains primarily stem from the introduction of coordinate adaptation, rather than merely increasing model capacity.

5 Insights

ACT delivers strong gains across architectures and datasets, which naturally raises a mechanistic question: what exactly are the ACT blocks doing inside the network? To better understand this, we visualize the layer-wise feature evolution and the effect of adding only the terminal ACT block in Figure 3. These visualizations suggest that ACT does not play a uniform role across depth. Instead, different blocks appear to divide their labor: earlier blocks mainly reorganize the latent representation, whereas the terminal block is more directly tied to the recovery of sharp local structure. Below, we unpack these two roles in turn.

5.1 How Intermediate ACT Blocks Simplify the Latent Operator

Earlier ACT blocks primarily serve as geometric realignment modules: instead of forcing every latent operator update to model transport or deformation on a fixed grid, they absorb part of this burden into learned coordinate adaptation, so that subsequent operator updates act on hidden representations with more regular spatial structure. In this way, they mitigate spatial misalignment before it accumulates across depth.

To illustrate this effect, we consider a simple setting of an integral operator:

$$\mathcal{F} : u \mapsto v, \qquad v(x) = \int_{\Omega} K(x, y)\, u(y)\, dy. \tag{21}$$

Next, we examine how input-side and output-side coordinate transformations modify the kernel representation of this operator. Detailed derivations are deferred to Appendix D.

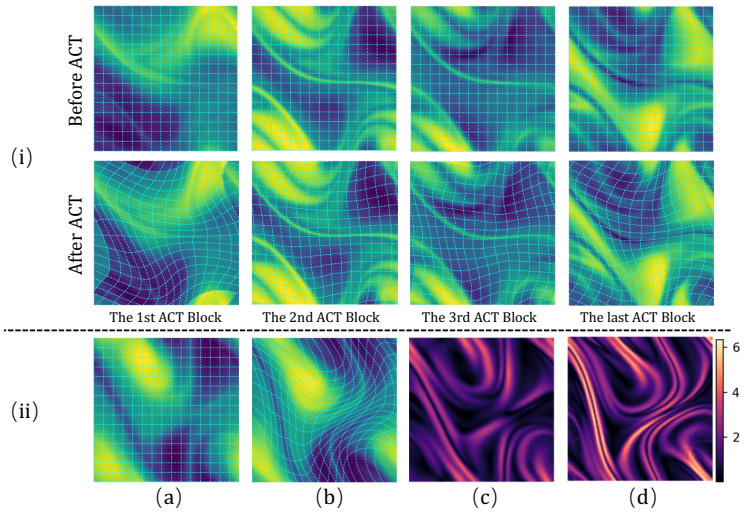

Figure 3: Visualization of feature representations induced by ACTs. (i) Layer-wise visualization of intermediate features, showing that the effect of ACT varies across depth, with both the first and the terminal blocks producing clear realignment. (ii) Features and gradient norms before and after adding only the final ACT block, showing that even this isolated terminal realignment sharpens local structure. Together, the two panels provide qualitative evidence that ACT first simplifies the latent operator mapping and then uses the final block to recover sharp detail.

Input-side transformation. For an input-side coordinate transformation written as $y = \phi_{\text{in}}(\xi)$, define the transformed input by $\bar{u}(\xi) = u\big(\phi_{\text{in}}(\xi)\big)$. The corresponding effective kernel is

$$\tilde{K}_{\text{in}}(x, \xi) = K\big(x, \phi_{\text{in}}(\xi)\big)\ \left|\det \nabla \phi_{\text{in}}(\xi)\right|, \tag{22}$$

Thus, input-side transformations re-parameterize the integration variable and change how information is aggregated from the input domain.

Output-side transformation. For an output-side coordinate transformation written as $x = \phi_{\text{out}}(\xi)$, define the transformed output by $\bar{v}(\xi) = v\big(\phi_{\text{out}}(\xi)\big)$. The effective kernel becomes

$$\tilde{K}_{\text{out}}(\xi, y) = K\big(\phi_{\text{out}}(\xi), y\big). \tag{23}$$

Hence, the transformation re-parameterizes the evaluation variable and changes where the operator is queried.

Summary. Input-side and output-side transformations therefore modify the operator in complementary ways: one changes the aggregation coordinates, while the other changes the query coordinates. In both cases, the original operator $\mathcal{F}$ is rewritten in transformed coordinates as an equivalent operator with a modified kernel representation. This is the mechanism by which geometric realignment can simplify the operator seen by subsequent latent updates.

5.2 How the Final ACT Block Restores Sharp Local Structure

Once earlier ACT blocks have reduced spatial misalignment in the latent features, the final block can focus on restoring sharp local structure near the output. This is consistent with the visualization above that ACT exhibits a depth-dependent division of labor: earlier ACT blocks mainly simplify the latent operator mapping, whereas the last block is important for reconstructing local detail.

Panel (ii) of Figure 3 provides a more direct check of this idea by isolating the case in which only the final ACT block is added. Even in this restricted setting, the final block visibly sharpens the feature representation and its gradient structure, providing qualitative evidence that terminal realignment contributes directly to recovering sharp local transitions.

One plausible mechanism for this sharpening effect is local coordinate compression. If the final ACT block induces an output-side re-parameterization $\bar{v}(\xi) = v\big(\phi(\xi)\big)$, then the chain rule gives

$$\nabla_\xi \bar{v}(\xi) = \nabla\phi(\xi)^\top \nabla_x v\big(\phi(\xi)\big). \tag{24}$$

Therefore, when ϕ locally compresses coordinates along a certain direction, the Jacobian matrix $\nabla\phi(\xi)$ acts as a multiplier on the physical gradient and can increase the magnitude of the corresponding derivative in the transformed representation. This provides a simple explanation for why the terminal ACT block can recover sharper local transitions.

The ablation results in Table 3 further support this interpretation. Adding only a final ACT already improves performance, while layer-wise ACT yields further gains beyond both the vanilla backbone and a larger-capacity baseline. Together, these results suggest that ACT distributes its effect across depth: intermediate blocks reduce geometric complexity, whereas the final block is especially important for recovering local sharpness.

6 Conclusion

In this work, we introduced ACT as a lightweight and plug-and-play mechanism for geometric realignment in neural operator learning. Across multiple backbones and PDE benchmarks, the results show that adaptive coordinate transformations can substantially improve accuracy, especially when transport, mixing, or deformation is a dominant source of complexity. The qualitative analyses and ablations further suggest that ACT helps redistribute part of the geometric burden from the operator updates into the learned coordinate system.

Several directions remain open for future work. In particular, it is important to better understand the theoretical conditions under which learned coordinate realignment simplifies operator learning, and to extend the current framework beyond the present structured-grid setting to more general boundary conditions, complex geometries, and irregular meshes. Integration with state-space neural operators is another future direction.

More broadly, our results suggest that coordinate choice can fundamentally shape the difficulty of operator learning, and the geometric realignment achieved by ACT offers a promising way to reduce this complexity.

Limitations: Despite these encouraging results, several aspects of ACT deserve further discussion. First, both the benchmark results and the analyses suggest that ACT is most beneficial when spatial misalignment, transport, mixing, deformation, or sharp moving interfaces are major sources of difficulty. In such settings, geometric realignment can simplify the latent operator while the terminal ACT block helps recover local sharpness. By contrast, for problems whose dynamics are already well represented in fixed Eulerian coordinates or do not exhibit pronounced geometric variability or sharp transitions, the improvement from ACT may be less significant.

Second, although our visualizations and ablation results support the interpretation that adaptive coordinate transformations simplify the effective operator seen by subsequent latent updates, this claim is currently supported mainly by empirical evidence rather than by a formal theoretical characterization. A more rigorous understanding of when and why learned coordinate realignment improves operator learning remains an important direction for future work.

References

Matthew W. Abruzzo, Drummond B. Fielding, and Greg L. Bryan. Taming the TuRMoiL: The Temperature Dependence of Turbulence in Cloud–Wind Interactions. *The Astrophysical Journal*, 966(2):181, May 2024. doi: 10.3847/1538-4357/ad1e51. 17

Guha Balakrishnan, Amy Zhao, Mert R Sabuncu, John Guttag, and Adrian V Dalca. Voxelmorph: a learning framework for deformable medical image registration. *IEEE transactions on medical imaging*, 38(8):1788–1800, 2019. 3

Andrey Bryutkin, Jiahao Huang, Zhongying Deng, Guang Yang, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. Hamlet: Graph transformer neural operator for partial differential equations. In *International Conference on Machine Learning*, pages 4624–4641. PMLR, 2024. 1

Jungyeon Cho and A. Lazarian. Compressible Magnetohydrodynamic Turbulence: mode coupling, scaling relations, anisotropy, new regime and astrophysical implications. *MNRAS*, 345:325–339, jan 2003. 18

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 3, 5

Bob D De Vos, Floris F Berendsen, Max A Viergever, Marius Staring, and Ivana Išgum. End-to-end unsupervised deformable image registration with a convolutional neural network. In *International Workshop on Deep Learning in Medical Image Analysis*, pages 204–212. Springer, 2017. 3

Nicki Skafte Detlefsen, Oren Freifeld, and Søren Hauberg. Deep diffeomorphic transformer networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4403–4412. IEEE, 2018. 3

Barath Ezhilan, Michael J Shelley, and David Saintillan. Instabilities and nonlinear dynamics of concentrated active suspensions. *Physics of Fluids*, 25(7), 2013. 17

Drummond B. Fielding, Eve C. Ostriker, Greg L. Bryan, and Adam S. Jermyn. Multiphase Gas and the Fractal Nature of Radiative Turbulent Mixing Layers. *The Astrophysical Journal Letters*, 894 (2):L24, May 2020. doi: 10.3847/2041-8213/ab8d2c. 17

Oren Freifeld, Søren Hauberg, Kayhan Batmanghelich, and Jonn W Fisher. Transformations based on continuous piecewise-affine velocity fields. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2496–2509, 2017. 3

Tong Gao, Meredith D Betterton, An-Sheng Jhang, and Michael J Shelley. Analytical structure, dynamics, and coarse graining of a kinetic model of an active fluid. *Physical Review Fluids*, 2(9): 093302, 2017. 17

Niccolò Grillo, James Rowbottom, Pietro Liò, Carola Bibiane Schönlieb, and Stefania Fresca. Hyper adaptivity: Joint hr-adaptive meshing via hypergraph multi-agent deep reinforcement learning. *arXiv preprint arXiv:2512.10439*, 2025. 3

Max Gronke and S. Peng Oh. The growth and entrainment of cold gas in a hot wind. *Monthly Notices of the Royal Astronomical Society: Letters*, 480(1):L111–L115, October 2018. doi: 10.1093/mnrasl/sly131. 17

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel De Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024. 19, 20, 30

Cyrill W Hirt, Anthony A Amsden, and JL Cook. An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of computational physics*, 14(3):227–253, 1974. 2, 4

Peiyan Hu, Yue Wang, and Zhi-Ming Ma. Better neural PDE solvers through data-free mesh movers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hj9ZuNimRl. 2, 3

Weizhang Huang and Robert D Russell. *Adaptive moving mesh methods*, volume 174. Springer Science & Business Media, 2010. 2, 4

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015. 3

David I. Ketcheson, Kyle T. Mandli, Aron J. Ahmadia, Amal Alghamdi, Manuel Quezada de Luna, Matteo Parsani, Matthew G. Knepley, and Matthew Emmett. PyClaw: Accessible, Extensible, Scalable Tools for Wave Propagation Problems. *SIAM Journal on Scientific Computing*, 34(4): C210–C231, November 2012. 16

G.A. Klaasen and W.C. Troy. Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, 44(1): 96–110, 1984. doi: 10.1137/0144008. 15

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. 1

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO. 1, 2, 6, 15, 22

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, 2023a. 1, 3

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, 2023b. URL https://arxiv.org/abs/2111.03794. 1

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 1, 2, 6, 20

Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*. Springer Science & Business Media, 2012. 20

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(3):218–229, 2021. 1

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=AM7iAhOkrx. 6

Suryanarayana Maddu, Scott Weady, and Michael J Shelley. Learning fast, accurate, and stable closures of a kinetic theory of an active fluid. *Journal of Computational Physics*, page 112869, 2024. 16

F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1 edition, 2016. doi: 10.1007/978-3-319-16874-6. 16

Till Muser, Alexandra Spitzer, Matti Lassas, Maarten V de Hoop, and Ivan Dokmanić. Flowers: A warp drive for neural pde solvers. *arXiv preprint arXiv:2603.04430*, 2026. 3, 27, 30, 32

Josephine Elisabeth Oettinger, Zakhar Shumaylov, Johannes Bostelmann, Jan Lellmann, and Carola-Bibiane Schönlieb. Diffeomorphism-equivariant neural networks, 2026. URL https://arxiv.org/abs/2602.06695. 3

Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina J Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B Dalziel, Drummond B Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024. 6, 16, 17, 18

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019. 2

James Rowbottom, Georg Maierhofer, Teo Deveney, Eike Hermann Müller, Alberto Paganini, Katharina Schratz, Pietro Lio, Carola-Bibiane Schönlieb, and Chris Budd. G-adaptivity: optimised graph-based mesh relocation for finite element methods. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=pyIXyl4qFx. 3

Zakhar Shumaylov, Peter Zaika, James Rowbottom, Ferdia Sherry, Melanie Weber, and Carola-Bibiane Schönlieb. Lie algebra canonicalization: Equivariant neural operators under arbitrary lie groups. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7PLpiVdnUC. 3

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35:1596–1611, 2022a. 6, 15, 16

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35:1596–1611, 2022b. 19

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2. 16

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries, 2024. URL https://arxiv.org/abs/2402.02366. 1, 2, 6

Niloufar Zakariaei, Siddharth Rout, Eldad Haber, and Moshe Eliasof. Advection augmented convolutional neural networks. *Advances in Neural Information Processing Systems*, 37:35225–35256, 2024. 3, 27, 30, 32

Mingrui Zhang, Chunyang Wang, Stephan Kramer, Joseph G Wallwork, Siyi Li, Jiancheng Liu, Xiang Chen, and Matthew D Piggott. Towards universal mesh movement networks. *Advances in Neural Information Processing Systems*, 37:14934–14961, 2024a. 3

Xuan Zhang, Jacob Helwig, Yuchao Lin, Yaochen Xie, Cong Fu, Stephan Wojtowytsch, and Shuiwang Ji. Sinenet: Learning temporal dynamics in time-dependent partial differential equations, 2024b. URL https://arxiv.org/abs/2403.19507. 2

Zhiwei Zhao, Changqing Liu, Yingguang Li, Zhibin Chen, and Xu Liu. Diffeomorphism neural operator for various domains and parameters of partial differential equations. *Communications Physics*, 8(1):15, 2025. 3

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable {detr}: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=gZ9hCDWe6ke. 3, 5

Appendix

A Additional Implementation Details

This appendix records implementation details that complement, rather than repeat, the ACT formulation in Section 3.3. In the current implementation, the backbone is instantiated as a 2D Fourier neural operator. The input is lifted by two pointwise convolutions into a hidden representation, processed by a stack of Fourier blocks, and then projected back to the output channels by another pointwise head. Each Fourier block consists of a spectral convolution, a skip connection, optional normalization, and an optional channel-mixing MLP. The implementation also supports gradient checkpointing on the Fourier blocks and an ablation mode in which ACT is disabled by replacing it with a pass-through module.

At the implementation level, each ACT block first applies a 1×1 projection to produce head-wise value features, then folds the head dimension into the batch dimension so that all heads are processed in parallel. If $x^{(l)}$ denotes the hidden feature map at layer l, the packed head features can be written as

$$V^{(l)} = \mathrm{reshape}\big(W_v x^{(l)}\big) \in \mathbb{R}^{(BM)\times C_h \times H \times W}. \tag{25}$$

When coordinate conditioning is enabled, the cached identity-grid channels are concatenated to these head-wise value features before offset prediction. In the default branched setting, the displacement tower combines parallel 1×1, 3×3, and dilated 3×3 convolutions before a final 1×1 fusion layer outputs the offsets. The resulting displacement is bounded by

$$\Delta^{(l)} = \alpha \, \tanh\Big(f_{\mathrm{disp}}\big([V^{(l)}, X_{\mathrm{id}}]\big)\Big), \tag{26}$$

where α is the prescribed maximum displacement. The last fusion layer is initialized to zero, so each ACT block starts exactly from the identity transformation. The code also includes a non-branched alternative in which the displacement tower is replaced by a two-layer 3×3 stack with GroupNorm and GELU.

The reference coordinates are cached as a normalized identity grid on $[-1, 1]^2$ and reused across calls as long as the spatial resolution and device do not change. The maximum displacement can either be specified manually or derived from the spatial resolution as the normalized size of one grid cell.

Resampling is carried out with bilinear `grid_sample` on normalized coordinates. If $G^{(l)} = X_{\mathrm{id}} + \Delta^{(l)}$ denotes the deformed sampling grid, the code applies a periodic fold only where coordinates are strictly outside the valid range,

$$\widetilde{G}^{(l)} = \begin{cases} G^{(l)}, & \text{if } G^{(l)} \in [-1, 1]^2, \\ \mathrm{remainder}\big(G^{(l)} + 1, 2\big) - 1, & \text{otherwise.} \end{cases} \tag{27}$$

Hence coordinates already equal to ± 1 remain unchanged, so zero predicted displacement reproduces the exact identity sampling pattern. The actual sampling call uses `padding_mode=border` and `align_corners=True`.

After resampling, the transformed head features are concatenated and projected back to the hidden width, then merged with a residual shortcut,

$$h^{(l+1)} = \begin{cases} \mathrm{Norm}\Big(W_o \, \widehat{V}^{(l)} + W_{\mathrm{res}} x^{(l)}\Big), & \text{for non-final ACT blocks,} \\ W_o \, \widehat{V}^{(l)} + W_{\mathrm{res}} x^{(l)}, & \text{for the final ACT block,} \end{cases} \tag{28}$$

where $\widehat{V}^{(l)}$ denotes the concatenated sampled head features. This matches the implementation flag that skips output normalization in the last ACT block. A further training-related detail is that displacement tensors are not returned in training mode: each ACT block outputs `(tensor, None)` while `self.training` is true, and only exposes the displacement fields in evaluation mode. This design reduces the need to retain layer-wise displacement tensors during training.

B Dataset Ilustrations

Table 4 provides a comprehensive overview of the datasets utilized in our experiments. We evaluate our models across a diverse suite of physical systems, encompassing both two-dimensional and

three-dimensional Cartesian coordinates. The table details the fundamental properties of each dataset, including the spatial resolution and the temporal length of each trajectory (`n_steps`). Furthermore, to ensure rigorous and reproducible benchmarking, we explicitly outline the data partitioning strategy, providing the exact number of trajectories allocated for the training, validation, and test splits.

Table 4: Detailed description of the datasets, including coordinate system (CS), spatial resolution, number of time-steps per trajectory (`n_steps`), and the number of trajectories in the training, validation, and test sets. DarcyFlow, Helmholtz, and Poisson-Gauss are steady-state operator-learning problems; each sample is encoded as a two-step pseudo-trajectory (input coefficient/source field at t=0, steady-state solution at t=1), hence `n_steps` $= 2$.

Dataset	CS	Resolution	n_steps	Train	Valid	Test
Navier_Stokes	(x, y)	128×128	45	400	50	50
Diff_Reaction	(x, y)	128×128	101	800	100	100
Shallow_Water	(x, y)	128×128	101	800	100	100
Active_Matter	(x, y)	256×256	81	175	24	26
TRL_2D	(x, y)	128×384	101	72	9	9
MHD_64	(x, y, z)	$64 \times 64 \times 64$	100	77	10	10
DarcyFlow	(x, y)	128×128	2	8000	1000	1000
Helmholtz	(x, y)	128×128	2	8000	1000	1000
Poisson-Gauss	(x, y)	128×128	2	8000	1000	1000

Navier-Stokes Equation: Our benchmark includes the two-dimensional Navier-Stokes (NS) equations governing a viscous and incompressible fluid. The dynamics are expressed via the vorticity formulation on a unit torus domain:

$$\begin{aligned} \partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) &= \nu \Delta w(x,t) + f(x), && x \in (0,1)^2, t \in (0,T] \\ \nabla \cdot u(x,t) &= 0, && x \in (0,1)^2, t \in [0,T] \\ w(x,0) &= w_0(x), && x \in (0,1)^2 \end{aligned} \tag{29}$$

In these governing equations, $u \in C([0,T]; H^r_{\text{per}}((0,1)^2; \mathbb{R}^2))$ (with $r > 0$) defines the fluid velocity, while $w = \nabla \times u$ denotes the scalar vorticity field. The initial condition is given by $w_0 \in L^2_{\text{per}}((0,1)^2; \mathbb{R})$, $\nu \in \mathbb{R}_+$ controls the kinematic viscosity, and $f \in L^2_{\text{per}}((0,1)^2; \mathbb{R})$ acts as a time-independent external forcing term. Following the data-generation procedure of FNO [Li et al., 2021], we use initial conditions and a fixed forcing

$$w_0 \sim \mathcal{N}\left(0,\, 7^{3/2}(-\Delta + 49I)^{-2.5}\right), \quad f(x) = 0.1\left(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2))\right).$$

We aim to approximate the solution operator $G^\dagger : \omega(\cdot, t_0 : t_k) \to \omega(\cdot, t_{k+1})$.

We evaluate the models under a representative viscosity regime with $\nu = 10^{-4}$. The data are generated at 512×512 and are downsampled to 128×128. The resulting trajectories used for training and testing have spatial resolution 128×128.

2D Diffusion-Reaction Equation : Our study also employs the two-dimensional diffusion-reaction datasets provided by PDEBench [Takamoto et al., 2022a]. This benchmark broadens the physical modeling to a 2D spatial domain and involves two non-linearly coupled quantities: the activator $u = u(t, x, y)$ and the inhibitor $v = v(t, x, y)$. The system's dynamics are governed by the following partial differential equations:

$$\partial_t u = D_u \partial_{xx} u + D_u \partial_{yy} u + R_u, \quad \partial_t v = D_v \partial_{xx} v + D_v \partial_{yy} v + R_v\,, \tag{30}$$

Here, D_u and D_v represent the constant diffusion coefficients associated with the activator and inhibitor, respectively. Meanwhile, $R_u = R_u(u, v)$ and $R_v = R_v(u, v)$ correspond to their localized reaction terms. The spatio-temporal domain is bounded by $x, y \in (-1, 1)$ and spans a time interval of $t \in (0, 5]$. Such coupled systems are widely recognized for their utility in simulating biological pattern generation. Specifically, the reaction mechanisms are modeled after the classic Fitzhugh-Nagumo kinetics [Klaasen and Troy, 1984], which are formulated as:

$$R_u(u, v) = u - u^3 - k - v, \tag{31}$$

$$R_v(u, v) = u - v, \tag{32}$$

For these experiments, the scalar parameter is fixed at $k = 5 \times 10^{-3}$, and the respective diffusion rates are specified as $D_u = 1 \times 10^{-3}$ and $D_v = 5 \times 10^{-3}$. Initial states are synthesized by sampling from a standard normal distribution, such that $u(0, x, y) \sim \mathcal{N}(0, 1.0)$ across the spatial grid. While the native PDEBench repository supplies high-resolution data with grid dimensions of $N_x = 512$, $N_y = 512$, and $N_t = 501$, we adopt the spatially and temporally downsampled variant ($N_x = 128$, $N_y = 128$, $N_t = 101$) to facilitate efficient neural network training. To generate the ground truth, the spatial derivatives were discretized via the finite volume method [Moukalled et al., 2016], while temporal progression was handled by the standard fourth-order Runge-Kutta (RK4) integrator available in the *scipy* library [Virtanen et al., 2020]. Overall, the core objective is to learn the solution operator $G^\dagger$ that maps the coupled historical states of the activator and inhibitor up to an initial time window to their subsequent evolution up to the next timestep. Formally, this operator we aim to learn is defined as $G^\dagger : C([t_0 : t_k]; H^r((-1, 1)^2; \mathbb{R}^2)) \to C(t_{k+1}; H^r((-1, 1)^2; \mathbb{R}^2))$.

Shallow Water Equations: For the shallow-water experiments, we adopt the benchmark configurations provided by PDEBench [Takamoto et al., 2022a]. Formulated as a depth-averaged simplification of the compressible Navier-Stokes equations, the shallow-water equations (SWE) serve as a fundamental mathematical model for simulating free-surface fluid dynamics. Mathematically, this 2D hyperbolic PDE system is naturally represented in a vectorized conservation form:

$$\partial_t h + \nabla \cdot (h\mathbf{u}) = 0, \tag{33a}$$

$$\partial_t (h\mathbf{u}) + \nabla \cdot \left(h\mathbf{u} \otimes \mathbf{u} + \frac{1}{2} g_r h^2 \mathbb{I} \right) = -g_r h \nabla b, \tag{33b}$$

In this notation, $\mathbf{u} = (u, v)$ captures the two-dimensional velocity vector, h stands for the localized water column depth, and b denotes the underlying bottom topography (bathymetry). Consequently, the product $h\mathbf{u}$ signifies the momentum vector, while g_r indicates the constant of gravitational acceleration. A critical property of this formulation is its strict preservation of mass and momentum conservation, even in the presence of discontinuous shock waves. This trait makes it highly suitable for generating complex, non-linear physical datasets that mirror real-world scenarios such as flash floods and tsunamis. By decomposing the vectors along the spatial axes, the governing dynamics can be explicitly formulated as:

$$\partial_t h + \partial_x hu + \partial_y hv = 0\,, \tag{34a}$$

$$\partial_t hu + \partial_x \left(u^2 h + \frac{1}{2} g_r h^2 \right) + \partial_y uvh = -g_r h \partial_x b\,, \tag{34b}$$

$$\partial_t hv + \partial_y \left(v^2 h + \frac{1}{2} g_r h^2 \right) + \partial_x uvh = -g_r h \partial_y b\,, \tag{34c}$$

Regarding the specific experimental setup, the PDEBench data collection models a two-dimensional radial dam-break phenomenon. The simulation operates within a bounded square region $\Omega = [-2.5, 2.5]^2$. The initial fluid depth is structured as a centralized cylindrical column, governed by the following piecewise function:

$$h(t = 0, x, y) = \begin{cases} 2.0, & \text{for } r < \sqrt{x^2 + y^2} \\ 1.0, & \text{for } r \geq \sqrt{x^2 + y^2} \end{cases} \tag{35}$$

Here, the initial column radius r is drawn stochastically from a uniform continuous distribution $\mathcal{U}(0.3, 0.7)$. The reference trajectories for these complex fluid interactions were computed via the robust finite volume schemes implemented within the PyClaw software library [Ketcheson et al., 2012]. Specifically, the primary objective is to approximate the solution operator $G^\dagger$ that maps the historical states of the water depth up to an initial time window to its future dynamics up to the next timestep, the operator we aim to learn is defined as $G^\dagger : C([t_0 : t_k]; H^r(\Omega; \mathbb{R})) \to C(t_{k+1}; H^r(\Omega; \mathbb{R}))$, tracking the temporal transition $h|_{\Omega \times [t_0 : t_k]} \mapsto h|_{\Omega, t_{k+1}}$, where Ω denotes the spatial domain.

Active Matter: We incorporate active matter datasets derived from the works of [Ohana et al., 2024, Maddu et al., 2024]. Fundamentally, active matter comprises autonomous entities—ranging from synthetic micro-swimmers to biological macromolecules—capable of transducing chemical energy into mechanical motion, consequently exerting local active stresses. Within the bulk, these generated forces propagate via mechanisms like hydrodynamic interactions, structural cross-linking,

or direct steric hindrances, resulting in intricately complex spatiotemporal behaviors. The specific configurations adopted in our benchmark model a suspension of active particles within a highly viscous medium. This environment induces orientation-coupled viscosity and pronounced long-range interactions, thereby presenting a formidable challenge for operator learning architectures and data-driven surrogate modeling.

The foundational data generation simulates N elongated active particles (with length ℓ and width b, satisfying $\ell/b \gg 1$) dispersed throughout a Stokes flow regime of volume V. As the system approaches the macroscopic limit, the suspension dynamics are effectively captured by continuum kinetic frameworks tracking the probability density function $\Psi(\mathbf{x}, \mathbf{p}, t)$ [Ezhilan et al., 2013, Gao et al., 2017]. The temporal evolution of Ψ, which inherently conserves the total particle count, is dictated by the Smoluchowski equation:

$$\frac{\partial \Psi}{\partial t} + \nabla_{\mathbf{x}} \cdot (\dot{\mathbf{x}}\Psi) + \nabla_{\mathbf{p}} \cdot (\dot{\mathbf{p}}\Psi) = 0, \tag{36}$$

Here, the variables $\dot{\mathbf{x}}$ and $\dot{\mathbf{p}}$ represent the translational and rotational fluxes, respectively, dictated by individual particle kinematics within the ambient velocity field $\mathbf{u}(\mathbf{x}, t)$. Integrating the distribution function over the orientational space ($\langle f \rangle = \int_{|\mathbf{p}|=1} f\Psi \, d\mathbf{p}$) allows us to extract macroscopic moments: the scalar concentration $c = \langle 1 \rangle$, the polarity vector $\mathbf{n} = \langle \mathbf{p} \rangle / c$, and the nematic order tensor $\mathbf{Q} = \langle \mathbf{pp} \rangle / c$. In the context of dense particle suspensions, these phase-space fluxes are formulated as:

$$\dot{\mathbf{x}} = \mathbf{u} - d_T \nabla_{\mathbf{x}} \log \Psi; \quad \dot{\mathbf{p}} = (\mathbf{I} - \mathbf{pp}) \cdot (\nabla \mathbf{u} + 2\zeta \mathbf{D}) \cdot \mathbf{p} - d_R \nabla_{\mathbf{p}} \log \Psi. \tag{37}$$

In these expressions, d_T and d_R denote the non-dimensional coefficients for translational and rotational diffusion, respectively. The parameter ζ quantifies the nematic alignment intensity driven by steric repulsion, while $\mathbf{D} = \langle \mathbf{pp} \rangle$ defines the second-order orientational moment. Furthermore, the fluid hydrodynamics are intricately coupled to the Smoluchowski dynamics via the incompressible Stokes equations:

$$-\Delta \mathbf{u} + \nabla P = \nabla \cdot \mathbf{\Sigma}, \quad \nabla \cdot \mathbf{u} = 0, \tag{38}$$

$$\mathbf{\Sigma} = \alpha \mathbf{D} + \beta \mathbf{S} : \mathbf{E} - 2\zeta\beta \left(\mathbf{D} \cdot \mathbf{D} - \mathbf{S} : \mathbf{D} \right). \tag{39}$$

Within this fluid coupling, $P(\mathbf{x}, t)$ stands for the hydrodynamic pressure. The parameters α and β scale the non-dimensional active dipole magnitude and the effective particle concentration, respectively. Additionally, $\mathbf{E} = [\nabla \mathbf{u} + \nabla \mathbf{u}^\top]/2$ calculates the symmetric strain-rate tensor, and $\mathbf{S} = \langle \mathbf{pppp} \rangle$ corresponds to the fourth-order orientational moment.

The coupled governing equations (36)-(38) were numerically resolved via a pseudo-spectral framework utilizing 256×256 spatial modes on a doubly periodic domain of size $(0, 10)^2$, integrated temporally with a second-order semi-implicit backward differentiation scheme (SBDF2). Key dimensionless parameters were systematically modulated: the active contractility coefficient was uniformly sampled from $\alpha \in \{-1, -2, -3, -4, -5\}$, the effective concentration parameter was fixed at $\beta = 0.8$, and the nematic alignment factor traversed $\zeta \in \{1, 3, 5, 7, 9, 11, 13, 15, 17\}$. The final curated dataset encapsulates an extensive suite of macroscopic physical fields: scalar concentration, 2D fluid velocity, orientational order tensor, and the symmetric strain-rate tensor. Formally, the target operator is defined as $G^\dagger : C([t_0 : t_k]; H^r_{\text{per}}((0,1)^2; \mathbb{R}^d)) \to C(t_{k+1}; H^r_{\text{per}}((0,1)^2; \mathbb{R}^d))$, tracking the temporal transition of the combined macroscopic state vector $\boldsymbol{\phi}$: $\boldsymbol{\phi}|_{(0,1)^2 \times [t_0:t_k]} \mapsto \boldsymbol{\phi}|_{(0,1)^2, t_{k+1}}$.

Turbulent Radiative Layer (2D): The benchmark additionally incorporates the 2D turbulent mixing layer dataset from The Well [Ohana et al., 2024], based on the foundational simulations by [Fielding et al., 2020]. This dataset models the interfacial dynamics between cold, dense gas and ambient hot plasma, where the survival of cold structures is dictated by the competition between turbulent mixing and rapid radiative cooling [Gronke and Oh, 2018, Abruzzo et al., 2024]. The primary objective is to learn the forward solution operator $G^\dagger : C([t_0 : t_k]; H^r(\Omega; \mathbb{R}^4)) \to C(t_{k+1}; H^r(\Omega; \mathbb{R}^4))$, which tracks the temporal transition of the coupled fluid fields across the 2D spatial domain Ω: $(\rho, P, \vec{v})|_{\Omega \times [t_0:t_k]} \mapsto (\rho, P, \vec{v})|_{\Omega, t_{k+1}}$.

Mathematically, the system governs the hydrodynamic conservation laws augmented with a temperature-dependent energy sink, parameterized by a characteristic cooling time t_{cool}:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{40}$$

$$\frac{\partial \rho \vec{v}}{\partial t} + \nabla \cdot (\rho \vec{v}\vec{v} + P) = 0 \tag{41}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + P)\vec{v}) = -\frac{E}{t_{\text{cool}}} \tag{42}$$

$$E = P/(\gamma - 1) \quad \text{where} \quad \gamma = 5/3 \tag{43}$$

A fundamental property of this system is the scaling relationship between the volume-integrated cooling rate, the net mass accretion flux, the relative velocity, and the cooling timescale, expressed as $\dot{E}_{\text{cool}} \propto \dot{M} \propto v_{\text{rel}}^{3/4} t_{\text{cool}}^{-1/4}$.

To systematically sample this dynamic landscape, the underlying high-fidelity simulations were generated by sweeping the cooling timescale across

$$t_{\text{cool}} \in \{0.03, 0.06, 0.1, 0.18, 0.32, 0.56, 1.00, 1.78, 3.16\}.$$

The finalized dataset captures the temporal evolution of the 2D density, pressure, and velocity fields, efficiently circumventing the substantial computational overhead required for conventional numerical integration.

Magnetohydrodynamic (MHD) Turbulence: We integrate the magnetohydrodynamic (MHD) turbulence dataset directly from The Well repository [Ohana et al., 2024]. Turbulence within MHD systems acts as a cornerstone for numerous astrophysical phenomena, dictating processes from interstellar medium (ISM) evolution to large-scale galactic formation. Specifically, this dataset features isothermal, non-self-gravitating MHD flows that accurately emulate the physical environment of the diffuse ISM. The primary objective is to approximate the solution operator $G^\dagger$ that maps the temporal history of the coupled magnetohydrodynamic fields—comprising the scalar density, the 3D fluid velocity vector, and the 3D magnetic field vector—up to an initial time window to their subsequent turbulent evolution up to a final time.

The governing ideal MHD equations, resolved via a third-order hybrid essentially non-oscillatory (ENO) scheme [Cho and Lazarian, 2003], formulate the conservation laws for mass, momentum, and magnetic induction as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0, \tag{44}$$

$$\frac{\partial \rho \boldsymbol{v}}{\partial t} + \nabla \cdot \left[\rho \boldsymbol{v}\boldsymbol{v} + \left(p + \frac{B^2}{8\pi} \right) \mathbf{I} - \frac{1}{4\pi} \mathbf{B}\mathbf{B} \right] = \mathbf{f}, \tag{45}$$

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla \times (\boldsymbol{v} \times \mathbf{B}) = 0. \tag{46}$$

In this system, ρ signifies the fluid density, $\boldsymbol{v}$ defines the local velocity vector, $\mathbf{B}$ represents the magnetic field, p denotes the isotropic gas pressure ($p = c_{\text{s}}^2 \rho$ with isothermal sound speed c_{s}), and $\mathbf{I}$ acts as the identity tensor. Continuous turbulence is sustained by the external source term $\mathbf{f}$, which operates as a stochastic solenoidal driving force peaking at the wavenumber $k \approx 2.5$.

To ensure a robust exploration of the chaotic parameter space, the dataset provides a diverse spectrum of turbulent flows systematically modulated across two key dimensionless parameters: the sonic Mach number ($\mathcal{M}_{\text{s}} \equiv |\boldsymbol{v}|/c_{\text{s}} \in \{0.5, 0.7, 1.5, 2.0, 7.0\}$) and the Alfvénic Mach number ($\mathcal{M}_{\text{A}} \equiv |\boldsymbol{v}|/\langle v_A \rangle \in \{0.7, 2.0\}$). These configurations encompass both magnetically-dominated sub-Alfvénic regimes and super-Alfvénic environments. The generated snapshots, curated at a 64^3 spatial resolution to balance fidelity with computational tractability, provide a highly comprehensive multi-modal dataset consisting of scalar mass density, vector fluid velocity, and vector magnetic fields. Formally, this operator is defined as $G^\dagger : C([t_0 : t_k]; H^r_{\text{per}}(\Omega; \mathbb{R}^7)) \to C(t_{k+1}; H^r_{\text{per}}(\Omega; \mathbb{R}^7))$, where Ω denotes the 3D periodic spatial domain, tracking the temporal transition $(\rho, \boldsymbol{v}, \mathbf{B})|_{\Omega \times [t_0, t_k]} \mapsto (\rho, \boldsymbol{v}, \mathbf{B})|_{\Omega, t_{k+1}}$, where Ω denotes the 3D periodic spatial domain.

Steady-state operator-learning benchmarks. In addition to the time-dependent datasets above, we evaluate on three steady-state elliptic problems in which the solution operator maps a spatially varying coefficient or source field to the corresponding equilibrium solution. These benchmarks differ from the time-dependent setting in several respects: the problems are time-independent, the operator maps coefficients rather than an initial state into a steady-state solution, and the boundary conditions are of Dirichlet type rather than periodic Herde et al. [2024]. To accommodate them within our temporal benchmark framework, each sample is encoded as a two-step pseudo-trajectory whose first frame holds the input field and whose second frame holds the solution, so that a single-step prediction with $\texttt{n_steps_input} = \texttt{n_steps_output} = 1$ recovers the steady-state operator exactly. All three datasets are discretized on a 128×128 Cartesian grid over the unit square.

DarcyFlow. We consider the steady-state Darcy flow over the unit square $D = (0,1)^2$ from PDEBench Takamoto et al. [2022b], in which the spatially varying diffusion coefficient $a(x)$ is the input of the system. The steady state is governed by

$$-\nabla \cdot \big(a(x)\nabla u(x)\big) = f(x), \qquad x \in D, \tag{47}$$

$$u(x) = 0, \qquad x \in \partial D, \tag{48}$$

where the forcing term f is set to a constant β that controls the scale of the solution; we use the $\beta = 1$ configuration. Rather than solving (47) directly, PDEBench obtains u by integrating the parabolic evolution equation

$$\partial_t u(x,t) - \nabla \cdot \big(a(x)\nabla u(x,t)\big) = f(x), \qquad x \in D, \tag{49}$$

from a random-field initial condition until a steady state is reached, using the same finite-difference scheme as its 1D diffusion-reaction benchmark. The learning task is the operator $a \mapsto u$ with homogeneous Dirichlet boundary conditions. The released file contains $10{,}000$ coefficient–solution pairs; we split them by sample in an $8{:}1{:}1$ ratio into $8{,}000$ training, $1{,}000$ validation, and $1{,}000$ test samples.

Helmholtz. The Helmholtz equation models wave propagation in the frequency domain. Following Poseidon Herde et al. [2024], we consider the variant

$$-\Delta u - \omega^2 a(x,y)\, u = 0, \qquad (x,y) \in D, \tag{50}$$

$$u(x,y) = b, \qquad (x,y) \in \partial D, \tag{51}$$

where $\omega = 5\pi/2$ is the frequency, $D = (0,1)^2$ is the domain, a is a spatially varying function describing the propagation medium, and b is a constant Dirichlet value prescribed on the whole boundary. The boundary value is drawn as $b \sim \mathcal{U}[0.25,\, 0.5]$. The medium a is a normalized superposition of a random number of Gaussians: an integer n is sampled uniformly from $\{2, \ldots, 7\}$, and for $1 \le i \le n$ the amplitudes, widths, and centers are drawn as $A_i \sim \mathcal{U}[0.5,\, 10]$, $\sigma_i \sim \mathcal{U}[0.05,\, 0.1]$, and $x_i, y_i \sim \mathcal{U}[0.2,\, 0.8]$. The unnormalized field is

$$\bar{a}(x,y) = -\sum_{i=1}^{n} A_i \exp\!\left(-\frac{(x_i - x)^2 + (y_i - y)^2}{2\sigma_i^2}\right), \qquad (x,y) \in (0,1)^2, \tag{52}$$

and the coefficient is obtained by min–max normalization, $a = (\bar{a} - \min \bar{a})/(\max \bar{a} - \min \bar{a})$. The solution operator maps the pair (a, b) to u. Solutions are computed with a finite-difference method at 128^2 resolution. Because the boundary value b varies across samples, we provide it to the network as an additional spatially constant input channel and learn the shifted target $\tilde{u} = u - b$, which satisfies homogeneous Dirichlet conditions and matches the boundary structure of the other two steady-state problems; the physical solution is recovered at evaluation time by adding b back. The original release comprises $19{,}675$ samples with a $19{,}035/128/512$ split; we use the first $10{,}000$ samples and split them $8{,}000/1{,}000/1{,}000$ into training, validation, and test sets, which yields a substantially larger test set than the official split.

Poisson-Gauss. We further consider the Poisson equation with homogeneous Dirichlet boundary conditions,

$$-\Delta u = f, \qquad (x,y) \in (0,1)^2, \tag{53}$$

$$u = 0, \qquad (x,y) \in \partial(0,1)^2, \tag{54}$$

Table 5: Detailed parameter breakdown for the ConvNeXt-UNet enhanced with the ACT module (ACT-CNextU). The ACT mechanism is composed of `MultiHeadACTBlocks` distributed across the encoder and decoder stages. Disabling them recovers the vanilla ConvNeXt-UNet.

Component	Params	Notes
Input / Stem		
Initial convolutions: Conv2d × 2	3,824	3,066 + 758
Vanilla ConvNeXt-UNet Architecture (Subtotals)		
Encoder (4 stages): Downsample + 2×ConvNeXt blocks	3,674,156	feature encoding
Bottleneck: 1×ConvNeXt block	3,651,648	latent representation
Decoder (4 stages): Conv2d + Upsample + 2×ConvNeXt blocks	11,241,934	feature decoding
Vanilla Backbone subtotal	**18,567,738**	multi-scale UNet
MultiHeadACTBlock (ACT, distributed across 8 stages)		apply ACT
value_proj Conv2d	485,888	channel mixing
disp branches (ModuleDict): 1x1, 3x3, dilated 3x3	22,080	$8\times(152+1304+1304)$
disp_fuse: GELU + Conv2d($24\to2$)	400	8×50
sampled with augmented grid	–	
out_proj Conv2d	485,730	project features to output
GroupNorm	3,780	output normalization
ACT modules subtotal	**997,878**	8 ACT blocks in total
Total trainable params (apply ACT)	**19,569,440**	UNetConvNextACT
Total trainable params (vanilla UNet)	**18,571,562**	`use_ACT=False`
Optional ACT overhead	**+997,878**	**+5.37%**

whose solution operator maps the source term f to the solution u. The source is a superposition of a random number of Gaussians,

$$f(x,y) = \sum_{i=1}^{N} \exp\left(-\frac{(x-\mu_{x,i})^2 + (y-\mu_{y,i})^2}{2\sigma_i^2}\right), \tag{55}$$

where $N \sim \mathrm{Geom}(0.4)$, $\mu_{x,i}, \mu_{y,i} \sim \mathcal{U}[0,\ 1]$, and $\sigma_i \sim \mathcal{U}[0.025,\ 0.1]$. The task therefore models the diffusion of a source composed of Gaussian bumps. The dataset was generated by Poseidon Herde et al. [2024] with a finite element method implemented in FEniCS Logg et al. [2012] and originally contains 20,000 samples with a $19{,}640/120/240$ split. As for Helmholtz, we use a 10,000-sample subset split $8{,}000/1{,}000/1{,}000$ into training, validation, and test sets.

C Model Structure and Hyperparameters

C.1 ACT-CNextU

As detailed in Table 5, the proposed Adaptive Coordinate Transformation (ACT) enhances the ConvNeXt-UNet[Liu et al., 2022] architecture by systematically integrating a `MultiHeadACTBlock` after each encoder and decoder stage. The vanilla ConvNeXt-UNet serves as a robust multi-scale backbone, comprising 18,571,562 trainable parameters. Enabling the ACT mechanism across all 8 hierarchical stages introduces an additional 997,878 parameters. This corresponds to an overhead of approximately 5.37% relative to the vanilla model, yielding a total of 19,569,440 parameters for the enhanced `UNetConvNextACT`. This marginal increase confirms that the performance improvements achieved by our method are fundamentally driven by the structured, coordinate-aware spatial transformations introduced by ACT, rather than a mere scaling of the model capacity.

We further emphasize that the integration of ACT is seamlessly adapted to the hierarchical nature of the U-Net architecture. Throughout all experimental evaluations, the underlying multi-scale encoder-bottleneck-decoder structure remains strictly fixed. While the offset generation branches within the ACT modules share a highly efficient, constant parameter footprint across stages, the linear projection layers naturally scale with the corresponding feature map dimensions and channel counts. Consequently, the computational cost introduced by the ACT modules is optimally distributed across the network's resolutions. This demonstrates that ACT is a highly efficient, plug-and-play enhancement that can be gracefully applied to modern hierarchical vision backbones without disrupting their original design principles.

Table 6: Detailed parameter breakdown for Transolver enhanced with the ACT module(ACT-Transolver). `MultiHeadACTBlock` acts as the ACT mechanism; disabling it recovers the vanilla Transolver architecture.

Component	Params	Notes
Input / Stem		
Global Parameter (Transolver root)	64	global token / state
TransolverMLP: Linear + GELU + Linear	9,664	1,408 + 8,256 (feature lifting)
Backbone stage (repeated 4 times)		
Step k ($k = 1 \dots 4$):	–	
TransolverBlock (single block details)		
LayerNorm	128	pre-normalization
PhysicsAttention1DEidetic	9,129	physics-attention
LayerNorm	128	pre-normalization
TransolverMLP: Linear + GELU + Linear	33,088	16,640 + 16,448
TransolverBlock subtotal	**42,473**	per block
MultiHeadACTBlock (when apply ACT)		apply ACT, if disabled, this becomes vanilla Transolver
value_proj Conv2d	16,640	$256 \to 64$
disp branches (ModuleDict): 1x1, 3x3, dilated 3x3	2,760	channel mixing
disp_fuse: GELU + Conv2d($24 \to 2$)	50	offset prediction
sampled with augmented grid	–	
out_proj Conv2d	16,448	project features to output channels
GroupNorm	128	output normalization
ACTBlock subtotal	**36,026**	per block
Backbone total (with ACT): $4\times$(TransolverBlock + ACTBlock)	313,996	$4\times(42,473{+}36,026)$
Backbone total (vanilla Transolver): $4\times$TransolverBlock	169,892	ACT disabled
Output Head for ACT-Transolver		
Projection: Conv2d + GELU + Conv2d	8,578	8,320 + 258
Output Head for Vanilla Transolver		
Projection: LayerNorm + Linear	258	
Total trainable params (apply ACT)	**332,302**	TransolverPlus
Total trainable params (vanilla Transolver)	**179,878**	`use_ACT=False`
Optional ACT overhead	**+144,104**	**+76.57%**

C.2 ACT-Translover

As detailed in Table 6, the proposed Adaptive Coordinate Transformation (ACT) augments the Transolver backbone by integrating a `MultiHeadACTBlock` after each standard `TransolverBlock`. Because the vanilla Transolver is an inherently lightweight architecture, comprising only 188,198 trainable parameters, enabling the ACT mechanism introduces an additional 144,104 parameters. Although this corresponds to a 76.57% relative parameter overhead, the absolute increase remains highly tractable, resulting in a total of just 332,302 parameters for the enhanced model (TransolverPlus). This demonstrates that the performance improvements achieved by our method are fundamentally driven by the structured, coordinate-aware feature transformations introduced by ACT, rather than relying on massive parameter scaling or heavy model capacities.

We further emphasize that, throughout all experimental evaluations, our Transolver-based models strictly adhere to the unified architectural configuration presented in Table 6. When deploying the model across different datasets, the core backbone architecture remains completely fixed. The only source of variation in the overall parameter count arises from the input lifting stem and the output projection head, whose dimensions depend solely on the dataset-specific input and output channels. Consequently, the parameter overhead introduced by the ACT blocks is essentially invariant across different datasets. Therefore, reporting a single representative parameter breakdown is sufficient to accurately characterize the architectural modifications and the additional computational cost introduced by ACT under all experimental settings.

Table 7: Layer-by-layer parameter breakdown of the proposed ACT-FNO architecture. The `MultiHeadACTBlock` acts as a lightweight, plug-and-play enhancement; removing it completely recovers the standard vanilla FNO baseline.

Component	Params	Notes
Input / Stem		
Lifting: Conv1x1($C_{in} \to 128$) + GELU + Conv1x1($128 \to 64$)	9,408	feature lifting
Backbone stage (repeated 4 times, alternating)		
Step k **(**$k = 1 \dots 4$**):**	–	
FNOBlock2d (single block details)		
SpectralConv2d(modes 32, channel 64)	4,194,304	dominant spectral weights
Conv2d (skip / linear)	4,096	channel mixing
GroupNorm $\times 2$	256	128 + 128
MLP: Conv1x1($64 \to 32$) + GELU + Conv1x1($32 \to 64$)	4,192	2,080 + 2,112
SoftGating	64	per-channel gate
FNOBlock2d subtotal	**4,202,912**	per block
MultiHeadACTBlock (when apply ACT)		apply ACT, if disabled, this becomes vanilla FNO
value_proj Conv1x1	4,160	$64 \to 64$
disp branches (ModuleDict): 1x1, 3x3, dilated 3x3	2,760	channel mixing
disp_fuse: GELU + Conv1x1($24 \to 2$)	50	offset prediction
sampled with arguemented grid	–	
out_proj Conv1x1	4,160	project features to output channels
GroupNorm	128	output normalization
ACTBlock subtotal	**11,258**	per block
Backbone total (with ACT): $4\times$(FNOBlock2d + ACTBlock)	16,856,680	$4 \times (4,202,912 + 11,258)$
Backbone total (vanilla FNO): $4\times$FNOBlock2d	16,811,648	ACT disabled
Head		
Projection: Conv1x1($64 \to 128$) + GELU + Conv1x1($128 \to C_{out}$)	8,578	8,320 + 258
Total trainable params (apply ACT)	**16,874,666**	ACT-FNO
Total trainable params (vanilla FNO)	**16,829,634**	`use_ACT=False`
Optional ACT overhead	**+45,032**	**+0.27%** only

C.3 ACT-FNO

Our NS experiments use a 16.8M-parameter FNO with 128×128 inputs, compared with approximately 0.4M parameters and 64×64 inputs in the original FNO-2D benchmark [Li et al., 2021]. We report one-step NRMSE, whereas the original benchmark reports trajectory-level error.

As summarized in Table 7, the proposed Adaptive Coordinate Transformation (ACT) augments the FNO backbone by inserting a lightweight `MultiHeadACTBlock` after each backbone stage, incurring only a marginal increase in trainable parameters. Concretely, enabling the ACT block introduces an additional 45,032 parameters, which corresponds to approximately 0.27% overhead relative to the vanilla FNO backbone. This clearly indicates that the performance improvements achieved by ACT are not attributed to increased model capacity, but rather to more effective and structured coordinate-aware feature transformations.

We further emphasize that, throughout all experimental evaluations, our FNO-based models strictly adhere to the unified architectural configuration presented in Table 7. When deploying the model across different datasets, the backbone architecture remains fixed, and the only source of variation in parameter count arises from the input lifting layer and the output projection head, whose dimensions depend solely on the dataset-specific input and output channels. Consequently, the relative parameter overhead introduced by the ACT block is essentially invariant across different datasets. Therefore, reporting a single representative parameter breakdown is sufficient to accurately characterize the additional computational cost introduced by ACT under all experimental settings.

Table 8: Common experimental settings shared by all compared models. Only the model architecture is changed across experiments, ensuring a fair comparison.

Category	Hyperparameter	Value
ACT Block	maximum displacement magnitude(Transolver&FNO)	0.5
	maximum displacement magnitude(Unet)	adaptive gird width($\frac{2}{\text{Resolution of Input}}$)
Data	Input steps	4
	Output steps	1
	Normalization	Z-score normalization
	Temporal stride	stride = 1
Batching	Batch size	32
Optimization	Optimizer	AdamW
	Learning rate	5×10^{-4}
	Weight decay	1×10^{-4}
LR schedule	Scheduler	StepLR
	Step size / decay factor for 2d data	50 / 0.5
	Step size / decay factor for 3d data	25 / 0.5
Training	Epochs for 2d data except `Active_Matter`	300
	Epochs for `Active_Matter`	200
	Epochs 3d data	100
	Loss function	MSE
	Gradient clipping	1.0
	Formatter	channels-first
	Checkpoint frequency	every 10 epochs
Validation	Validation frequency	every 20 epochs
	Short validation length	30 batches

C.4 Common Experiment Settings

To ensure a rigorous and fair evaluation, all compared models are trained and tested under a strictly unified experimental pipeline. The comprehensive list of common hyperparameters and configuration details is summarized in Table 8. Specifically, the data formulation is fixed to utilize 4 historical steps to predict the subsequent 1 step, with consistent Z-score normalization and temporal strides. Furthermore, the optimization dynamics—including the AdamW optimizer, learning rate schedules, batch size, and total training epochs—are held strictly constant across all baselines and our proposed models. Crucially, by keeping these training and validation protocols identical, we isolate the network architecture as the sole independent variable. This guarantees that any observed performance improvements are strictly attributable to the architectural enhancements (such as the integration of the ACT module) rather than dataset-specific hyperparameter tuning or training tricks.

C.5 Computing Configuration

As shown in table 9, all models and physical simulations were executed on a high-performance computing (HPC) cluster running a Linux-based operating system. The compute nodes are equipped with modern multi-core server-grade processors and large-memory configurations to support data-intensive workloads.

To accelerate neural network training and large-scale data generation, GPU-enabled nodes were utilized. Each node is equipped with four NVIDIA A100 Tensor Core GPUs, each providing 80 GB or 40 GB of VRAM, enabling efficient distributed and memory-intensive computations.

Table 9: Hardware configuration of the GPU-enabled compute nodes used for model training.

Component	Specification
Operating System	Linux-based HPC environment
CPU	Multi-core server-grade processors (node-dependent)
GPU	$4\times$ NVIDIA A100 (80 GB VRAM or 40 GB VRAM)
System Memory	High-memory configuration (node-dependent)

To evaluate the computational efficiency of the proposed method, Table 2 reports the number of learnable parameters, computational cost in GFLOPs, peak GPU memory consumption, and single-forward inference latency for the vanilla backbones and their ACT-augmented counterparts on the Shallow Water dataset. All measurements are conducted on NVIDIA A100 GPUs with a batch size of 32 and the same input resolution. The reported inference latency is the median over three measured batches after one warm-up batch, while GFLOPs are estimated using the same counting tool and input size for all models.

Interleaving ACT blocks introduces additional computational overhead because of the dynamic coordinate reparameterization and grid resampling operations. Nevertheless, the increase in model size remains limited for the convolutional and Fourier-based backbones. For example, ACT-FNO adds only 0.04M parameters, corresponding to a 0.2% increase over FNO, while increasing the inference time by 0.025 s per batch. Across the three architectures, the GFLOPs increase by 1.46–2.55×, and the inference latency increases by 1.64–1.89×. These overheads remain practical given the consistent accuracy gains achieved by ACT, which reduce the relative prediction errors by approximately 30% to more than 70% across the evaluated architectures.

D Derivation of Transformed Operator Representations

This appendix expands the discussion in Section 5.1 and shows explicitly how coordinate transformations modify the kernel representation of an integral operator. The notation follows Section 3: ψ_{in} and ψ_{out} denote forward coordinate transformations, while $\phi_{\mathrm{in}} := \psi_{\mathrm{in}}^{-1}$ and $\phi_{\mathrm{out}} := \psi_{\mathrm{out}}^{-1}$ denote the associated inverse maps. In the ACT instantiation of Section 3.3, these abstract transformations are realized implicitly through the layer-wise blocks $\mathcal{T}_\theta^{(l)}$ and their head-specific sampling maps $\phi_h^{(m)}$.

Consider

$$\mathcal{F} : u \mapsto v, \qquad v(x) = \int_\Omega K(x,y)\, u(y)\, dy. \tag{56}$$

Throughout this appendix, we assume that the coordinate transformations are bijective and sufficiently smooth so that the standard change-of-variables formula applies. The goal is not to change the underlying operator itself, but to rewrite the same mapping in different coordinates and make explicit how the kernel representation is modified.

Input-side transformation. Let $\psi_{\mathrm{in}} : \Omega \to \Omega$ be an input-side coordinate transformation, define transformed coordinates by $\xi = \psi_{\mathrm{in}}(y)$, and let $\phi_{\mathrm{in}} := \psi_{\mathrm{in}}^{-1}$ denote the associated inverse map. The transformed input is

$$\bar{u}(\xi) = u\big(\phi_{\mathrm{in}}(\xi)\big). \tag{57}$$

Equivalently, $u(y) = \bar{u}(\psi_{\mathrm{in}}(y))$, so the operator can be viewed as acting on the pulled-back input $\bar{u}$ instead of the original field u. Since only the source variable y is re-parameterized, the evaluation coordinate x remains unchanged. Applying the change of variables $y = \phi_{\mathrm{in}}(\xi)$, which leads to

$$dy = |\det \nabla \phi_{\mathrm{in}}(\xi)|\, d\xi, \tag{58}$$

where $\nabla \phi_{\mathrm{in}}$ is the Jacobian matrix of ϕ_{in}, and $\det$ denotes the determinant. Substituting this relation into the operator gives

$$\begin{aligned} v(x) &= \int_\Omega K(x,y)\, u(y)\, dy \\ &= \int_\Omega K\big(x, \phi_{\mathrm{in}}(\xi)\big)\, u\big(\phi_{\mathrm{in}}(\xi)\big)\ |\det \nabla \phi_{\mathrm{in}}(\xi)|\ d\xi \\ &= \int_\Omega K\big(x, \phi_{\mathrm{in}}(\xi)\big)\, \bar{u}(\xi)\ |\det \nabla \phi_{\mathrm{in}}(\xi)|\ d\xi \\ &= \int_\Omega \tilde{K}_{\mathrm{in}}(x,\xi)\, \bar{u}(\xi)\, d\xi, \end{aligned} \tag{59}$$

with the transformed kernel

$$\tilde{K}_{\mathrm{in}}(x,\xi) = K\big(x, \phi_{\mathrm{in}}(\xi)\big)\ |\det \nabla \phi_{\mathrm{in}}(\xi)|\,. \tag{60}$$

Therefore, an input-side transformation induces a new operator representation whose kernel incorporates two effects at once: the source location y is replaced by its expression in the transformed coordinates, and the Jacobian factor compensates for the change in volume element. This is why input-side transformations modify how information is aggregated over the source domain.

Output-side transformation. Let $\psi_{\text{out}} : \Omega \to \Omega$ be an output-side coordinate transformation, define transformed coordinates by $\xi = \psi_{\text{out}}(x)$, and let $\phi_{\text{out}} := \psi_{\text{out}}^{-1}$ denote the associated inverse map. The transformed output is

$$\bar{v}(\xi) = v\big(\phi_{\text{out}}(\xi)\big). \tag{61}$$

Here the situation is different from the input-side case: the integration variable y is untouched, so no Jacobian term appears. We simply evaluate the original output field at the physical point $x = \phi_{\text{out}}(\xi)$ corresponding to the transformed coordinate ξ. Substituting this relation into the operator gives

$$\bar{v}(\xi) = v\big(\phi_{\text{out}}(\xi)\big) = \int_{\Omega} K\big(\phi_{\text{out}}(\xi), y\big)\, u(y)\, dy. \tag{62}$$

This corresponds to the transformed kernel

$$\tilde{K}_{\text{out}}(\xi, y) = K\big(\phi_{\text{out}}(\xi), y\big). \tag{63}$$

Hence, an output-side transformation changes the representation of the operator only through the query variable. In other words, it does not alter how source contributions are integrated, but only where the operator is sampled.

Summary. The two transformations act on complementary variables of the integral operator. Input-side transformations modify how information is aggregated from the source domain, while output-side transformations modify where the operator is evaluated. In both cases, the underlying mapping is unchanged, but the kernel representation seen by the learner is altered.

E Additional Experimental Results

This appendix complements the main text with five groups of experiments on the Adaptive Coordinate Transform (ACT): autoregressive rollout accuracy, gradient and spectral fidelity, physical-consistency diagnostics, statistics of the learned coordinate maps, and steady (non-advective) elliptic problems. All quantities are defined in Section E.1 exactly as implemented in our evaluation code; the remaining subsections report results and discuss them. Together, these experiments examine whether feature-level coordinate realignment improves both predictive accuracy and the fidelity of physically relevant spatial structure across convolutional, spectral, and attention-based operator families.

E.1 Evaluation metrics

Notation. A prediction $\widehat{u}$ and reference u are compared on a uniform grid of N^d points ($d \in \{2, 3\}$) over a periodic domain $[0, L]^d$, with cell volume $\Delta V = (L/N)^d$ and wavenumbers $\mathbf{k} \in \{2\pi m/L : m \in \mathbb{Z}^d,\ |m_i| \le N/2\}$. Spatial means are written $\langle\cdot\rangle = N^{-d}\sum_{\mathbf{x}}(\cdot)$, and ε denotes a small constant for numerical stability (10^{-7} for NRMSE and 10^{-12} elsewhere). Unless stated otherwise, every metric is computed on a single spatial snapshot of a single field and then averaged over all fields, all time steps of the rollout, and all test trajectories. All diagnostics are computed in physical units after undoing the training-time normalization.

Prediction error. For a field u the normalized root-mean-square error is

$$\mathrm{NRMSE}(\widehat{u}, u) = \sqrt{\frac{\langle(\widehat{u} - u)^2\rangle}{\langle u^2\rangle + \varepsilon}}, \tag{64}$$

i.e. the root-mean-square error normalized by the root-mean-square magnitude of the reference field on the same snapshot. The per-snapshot values are averaged over fields, time steps, and trajectories. One-step NRMSE uses the ground-truth history as input at every step; rollout NRMSE uses the free autoregressive protocol of Eq. (70).

Spectral derivatives and velocity reconstruction. Derivatives on the periodic grid are evaluated spectrally, $\widehat{\partial_j f}(\mathbf{k}) = \mathrm{i}k_j\,\widehat{f}(\mathbf{k})$, where $\widehat{f}$ is the discrete Fourier transform. For the vorticity-form Navier–Stokes (NS) benchmark the model predicts the scalar vorticity ω, and the divergence-free velocity is reconstructed from the stream function,

$$\widehat{\psi} = \frac{\widehat{\omega}}{|\mathbf{k}|^2}, \qquad \widehat{u}_x = \mathrm{i}k_y\widehat{\psi}, \qquad \widehat{u}_y = -\mathrm{i}k_x\widehat{\psi}, \tag{65}$$

with the $\mathbf{k} = 0$ mode set to zero. For shallow water (SW) the diagnostics below are applied directly to the predicted height field h; for MHD_{64} they are applied to each of the predicted state components separately and then averaged over components.

Relative gradient error. With ∇ the spectral gradient,

$$\mathcal{E}_\nabla(\widehat{f}, f) = \frac{\sqrt{\langle \|\nabla\widehat{f} - \nabla f\|_2^2\rangle}}{\sqrt{\langle \|\nabla f\|_2^2\rangle} + \varepsilon}, \tag{66}$$

evaluated on ω for NS, on h for SW, and per component for MHD.

Shell-averaged spectrum and band errors. Fourier coefficients are normalized as $\widetilde{f} = \widehat{f}/N^d$ and binned into integer radial shells $q(\mathbf{k}) = \mathrm{round}\big(\|\mathbf{k}\|_2/(2\pi/L)\big)$. The shell spectrum is

$$E(q) = \sum_{\mathbf{k}:\, q(\mathbf{k})=q} \tfrac{1}{2}\,|\widetilde{f}(\mathbf{k})|^2, \tag{67}$$

where for NS the summand is the kinetic energy $\frac{1}{2}\big(|\widetilde{u}_x|^2 + |\widetilde{u}_y|^2\big)$ of the reconstructed velocity, and for SW and MHD it is the power of the scalar field under consideration. The available shells $q = 0, \ldots, Q$ (with $Q = \mathrm{round}(\sqrt{d}\,N/2)$, i.e. $Q = 91$ for 128^2 grids and $Q = 55$ for 64^3 grids) are divided into three contiguous bands of approximately equal width, $\mathcal{B}_{\mathrm{low}}$, $\mathcal{B}_{\mathrm{mid}}$, $\mathcal{B}_{\mathrm{high}}$; for 128×128 these are $q = 0$–29, 30–60, and 61–91, and for 64^3 they are $q = 0$–17, 18–36, and 37–55. Note that q is independent of L, so the bands are identical for NS ($L = 1$) and SW ($L = 5$). The relative spectral error in band $\mathcal{B}$ is

$$\mathcal{E}_{\mathcal{B}} = \frac{\big\|\big(\widehat{E}(q) - E(q)\big)_{q\in\mathcal{B}}\big\|_2}{\big\|\big(E(q)\big)_{q\in\mathcal{B}}\big\|_2 + \varepsilon}, \tag{68}$$

where $\widehat{E}$ and E are the shell spectra of prediction and reference. Because the reference energy in the high-frequency band is small, this quantity can exceed unity by orders of magnitude when a model injects spurious high-frequency content, which is the behavior observed for the vanilla Transolver in Table 11.

Conservation diagnostics. Integrals are discretized as $\int f\,\mathrm{d}V \approx \Delta V \sum_{\mathbf{x}} f$. All conservation errors are measured against the ground truth at the same rollout step rather than against the initial condition, because viscosity, forcing, or shocks make the exact invariants time dependent in these benchmarks.

- *NS total-vorticity drift.* Since $\int \omega\,\mathrm{d}A \approx 0$ for the initial conditions of this benchmark, a relative error is ill-conditioned; we therefore report the absolute discrepancy $\big|\int \widehat{\omega}\,\mathrm{d}A - \int \omega\,\mathrm{d}A\big|$.
- *NS enstrophy error.* The relative error of enstrophy $Z = \frac{1}{2}\int \omega^2\,\mathrm{d}A$, $|\widehat{Z} - Z|/(Z + \varepsilon)$.
- *SW mass and energy errors.* With $M = \int h\,\mathrm{d}A$ and $E = \frac{1}{2}\int h^2\,\mathrm{d}A$ on the $L = 5$ domain, we report $|\widehat{M} - M|/(|M| + \varepsilon)$ and $|\widehat{E} - E|/(E + \varepsilon)$. The quadratic quantity E is the L^2 energy of the height field and serves as a scale-sensitive integral invariant; it is not the full hydrodynamic energy of the shallow-water system.
- MHD_{64} *mass error.* With ρ the predicted density component, $M = \int \rho\,\mathrm{d}V$ and the relative error $|\widehat{M} - M|/(|M| + \varepsilon)$ is evaluated at each autoregressive lead time and averaged over trajectories.

Learned-map diagnostics. Each warp head h realizes $\phi_h(\xi) = \xi + \Delta_h(\xi)$ on the normalized coordinate grid $\xi \in [-1, 1]^2$ with spacing $\delta = 2/(N-1)$, where $\Delta_h = \alpha \tanh(\Delta_{h,\mathrm{raw}})$ is the bounded residual displacement. The displacement fields are captured by forward hooks on every ACT block during a free rollout. Per pixel we record the displacement norm $\|\Delta_h\|_2$ and the Jacobian determinant of the map,

$$\det J_h = \Big(1 + \frac{\partial \Delta_{h,x}}{\partial x}\Big)\Big(1 + \frac{\partial \Delta_{h,y}}{\partial y}\Big) - \frac{\partial \Delta_{h,x}}{\partial y}\,\frac{\partial \Delta_{h,y}}{\partial x}, \tag{69}$$

with partial derivatives approximated by second-order central differences (one-sided at the grid edges). The folding rate is the fraction of pixels with $\det J_h < 0$, i.e. locations where the map locally reverses orientation. Statistics are pooled over all heads, batch elements, and pixels for each block and rollout step; the global row of Table 13 averages the per-step statistics over blocks and steps.

E.2 Baseline models

Beyond the three backbone families (CNextU, FNO, and Transolver) that we evaluate with and without ACT, we compare against two standalone models that build spatial transport directly into the architecture.

Advection-Augmented CNNs. ADRNet [Zakariaei et al., 2024] incorporates a semi-Lagrangian transport mechanism into convolutional networks, using a learned velocity field to model advection alongside learned reaction and diffusion components. Unlike this explicit decomposition of physical dynamics, ACT uses learned deformation to realign latent features without prescribing a particular physical process, allowing it to generalize beyond advection-dominated problems. Since the publicly available implementation of ADRNet supports only two-dimensional data, its results on MHD_{64} are marked as unavailable.

FLOWERS. FLOWERS [Muser et al., 2026] employs learnable warping and multiscale feature interactions to model transport dynamics in PDEs. While both FLOWERS and ACT leverage learned spatial deformations, they differ in their design and scope. FLOWERS is a complete neural PDE solver architecture that incorporates deformation into its network design, whereas ACT employs deformation more generally as a layer-wise feature realignment mechanism, without restricting its application to advection-dominated dynamics. Moreover, ACT is a model-agnostic plug-in module that can be integrated into existing neural operator backbones without modifying their underlying architectures.

E.3 Autoregressive rollout performance

The time-dependent models predict the next state from four consecutive states. At test time, free rollout is seeded with the first four ground-truth states only and feeds every prediction back into the sliding history window:

$$\widehat{u}_{t+1} = \mathcal{F}_\theta(\widetilde{u}_{t-3}, \ldots, \widetilde{u}_t), \qquad \widetilde{u}_j = u_j \ (j < 4), \quad \widetilde{u}_j = \widehat{u}_j \ (j \geq 4). \tag{70}$$

We evaluate complete trajectories for NS and SW and an 8-step trajectory for the 64^3 magnetohydrodynamics benchmark (MHD_{64}). Because each prediction is returned to the input history, the rollout error measures how one-step errors propagate under repeated application of the operator. Table 10 reports NRMSE (Eq. (64)) for one-step predictions and for rollouts.

ACT improves every entry of Table 10: all twelve dataset–backbone–horizon combinations show lower error with ACT, with reductions ranging from 18% to over 99%. The gains are largest where the vanilla backbone struggles most. Vanilla Transolver diverges under free rollout on NS and SW (NRMSE of 1.02 and 0.48), whereas ACT-Transolver remains stable at 0.1864 and 0.00408, reductions of 81.8% and 99.2%; its one-step error on NS likewise drops from 0.3229 to 0.0135. For FNO, which is already a strong baseline, ACT still reduces the one-step NRMSE by 72.1% on NS and 41.0% on MHD_{64}, and the rollout error by 54.7% on SW and 18–19% on NS and MHD. ACT-Transolver attains the lowest one-step NRMSE on MHD_{64} (0.0682) and the lowest rollout NRMSE on MHD (0.351), while ACT-FNO is best on the remaining NS and SW rows. The convolutional backbone benefits as well: ACT-CNextU lowers the one-step NRMSE from 0.0613 to 0.0496 on NS, from 2.11×10^{-4} to 1.58×10^{-4} on SW, and from 0.1208 to 0.0963 on MHD (Table 15).

Table 10: **One-step and rollout performance.** NRMSE for FNO and Transolver on NS, SW, and MHD_{64}. NS/SW use complete trajectories; MHD uses an 8-step trajectory. Gray columns denote ACT-augmented models. Values marked with ↓ give the relative error reduction over the vanilla (VAN) model. Best results are in **bold**.

Dataset	Metric	FNO		Transolver	
		VAN	+ACT	VAN	+ACT
NS	One-step	0.0222	**0.0062 ↓72.1%**	0.3229	**0.0135 ↓95.8%**
	Rollout	0.1952	**0.1596 ↓18.2%**	1.0216	**0.1864 ↓81.8%**
SW	One-step	1.38×10^{-4}	$\mathbf{0.93 \times 10^{-4}}$ **↓32.6%**	3.49×10^{-4}	$\mathbf{0.78 \times 10^{-4}}$ **↓77.7%**
	Rollout	0.00660	**0.00299 ↓54.7%**	0.48400	**0.00408 ↓99.2%**
MHD_{64}	One-step	0.1540	**0.0909 ↓41.0%**	0.2930	**0.0682 ↓76.7%**
	Rollout	0.441	**0.358↓18.8%**	0.671	**0.351 ↓47.7%**

Table 11: **Gradient and spectral fidelity.** Relative gradient error and low-, mid-, and high-frequency spectral errors on NS/SW full rollouts and the 8-step MHD rollout (lower is better). Gray columns denote ACT-augmented models; values marked with ↓ give the relative error reduction over VAN. Best results in each dataset–metric row are in **bold**.

Dataset	Error ↓	FNO		Transolver	
		VAN	+ACT	VAN	+ACT
NS	Gradient	0.529	**0.447 ↓15.5%**	3.628	**0.483↓86.7%**
	Low-frequency	**0.010**	0.011	0.855	**0.011 ↓98.7%**
	Mid-frequency	1.066	**0.657 ↓38.4%**	3941.000	**0.744↓>99.9%**
	High-frequency	73.700	**4.950↓93.3%**	12675.000	**2.410 ↓>99.9%**
SW	Gradient	0.187	**0.070 ↓62.6%**	12.620	**0.074↓99.4%**
	Low-frequency	0.0010	**0.0006 ↓40.0%**	1.168	**0.0026↓99.8%**
	Mid-frequency	0.143	**0.065 ↓54.5%**	292.800	**0.089↓>99.9%**
	High-frequency	3.435	**0.099 ↓97.1%**	15175.000	**0.119↓>99.9%**
MHD_{64}	Gradient	0.849	**0.731↓13.9%**	1.262	**0.703 ↓44.3%**
	Low-frequency	0.132	**0.079 ↓40.2%**	0.393	**0.126↓67.9%**
	Mid-frequency	0.537	**0.209↓61.1%**	0.578	**0.141 ↓75.6%**
	High-frequency	0.783	**0.528↓32.6%**	4.090	**0.418 ↓89.8%**

Because the same block yields these gains in convolutional, spectral, and attention-based operators, we attribute them to the coordinate realignment itself rather than to any backbone-specific tuning; the rollout results in particular show that the improvement compounds rather than erodes under repeated application of the operator.

E.4 Gradient and spectral fidelity

Gradient and spectral errors test whether lower prediction error is accompanied by recovery of the target spatial structure. We evaluate FNO and Transolver with and without ACT using the relative gradient error (Eq. (66)) and the shell-averaged band spectral errors (Eq. (68)), computed on free autoregressive predictions (complete NS/SW rollouts and the 8-step MHD rollout) rather than on one-step outputs. Lower values indicate closer agreement with the reference gradients and spectrum.

ACT improves the gradient error and the mid- and high-frequency spectral errors in all six dataset–backbone comparisons of Table 11, and the low-frequency error in five of the six; the single exception (FNO on NS, 0.010 versus 0.011) is within the resolution of the metric. The high-frequency band, which is the most direct indicator of spectral bias, shows the largest effect: ACT reduces the high-band error of FNO by 93.3% on NS and 97.1% on SW, and of Transolver by more than 99.9% on both. For the vanilla Transolver the mid- and high-band errors reach 10^2–10^4 on NS and SW, i.e. the rollout accumulates spurious energy at scales that the reference solution does not contain; with ACT the same backbone attains mid- and high-band errors of 0.09–2.4, on the same scale as ACT-FNO, so the failure mode is eliminated rather than merely attenuated. Gradient fidelity improves in step with the spectrum: the relative gradient error drops by 62.6% for FNO on SW and by 86.7% and 99.4% for Transolver on NS and SW, and on the three-dimensional MHD_{64} benchmark ACT-Transolver reaches

Table 12: **Physical consistency diagnostics.** Total-vorticity drift, enstrophy, energy, and mass errors on NS, SW, and MHD_{64}, evaluated on the same full rollout predictions as Table 10 (lower is better). Gray columns denote ACT-augmented models. Best results in each row are in **bold**.

Dataset / metric	FNO		Transolver	
	VAN	+ACT	VAN	+ACT
NS total-vorticity drift ↓	**0.0009**	0.0020	0.598	**0.0017**
NS enstrophy error ↓	**0.010**	0.011	0.851	0.011
SW mass error ↓	0.0005	**0.0003**	0.449	**0.0013**
SW energy error ↓	0.0010	**0.0006**	1.171	**0.0027**
MHD_{64} mass error ↓	0.0189	**0.0184**	0.479	**0.0111**

the lowest gradient, mid-, and high-band errors of the table (0.703, 0.141, 0.418). The simultaneous improvement in gradient accuracy and in mid-/high-frequency spectral agreement shows that ACT recovers resolved spatial structure during autoregressive prediction, rather than producing visually sharper fields by injecting high-frequency content.

E.5 Physical consistency diagnostics

Interpolation-based resampling does not impose a discretely conservative stencil by construction. We therefore assess physical consistency empirically through the conservation diagnostics of Section E.1, evaluated on the same rollout predictions as above. Table 12 reports matched vanilla (VAN) and ACT-augmented models for FNO and Transolver; lower values indicate smaller drift or error.

The central observation of Table 12 is that interpolation-based resampling does not degrade physical consistency in practice and, for the global operators, improves it markedly. The effect is most pronounced for Transolver, whose vanilla version violates every diagnostic by a large margin (errors of 0.45–1.17): ACT reduces its NS total-vorticity drift from 0.598 to 0.0017, the NS enstrophy error from 0.851 to 0.011, the SW mass and energy errors from 0.449 and 1.171 to 0.0013 and 0.0027, and the MHD_{64} mass error from 0.479 to 0.0111, reductions of 97.7–99.8% that bring ACT-Transolver to within a small factor of the best model in every row. For FNO, which is already nearly conservative, ACT still lowers the SW mass and energy errors by 40% each ($0.0005 \to 0.0003$ and $0.0010 \to 0.0006$) and achieves the best SW values in the table, while the NS enstrophy error (0.010 versus 0.011) and MHD mass error (0.0189 versus 0.0184) are essentially unchanged; the only increase, in NS total-vorticity drift ($0.0009 \to 0.0020$), is an absolute quantity on the order of 10^{-3} for a field whose integral vanishes to leading order. We draw two conclusions. First, the empirical conservation errors of the ACT models are small in absolute terms across all backbones, so the lack of a discretely conservative stencil is not a practical limitation. Second, the gains are largest exactly where structures move or exhibit localized transitions, which is consistent with ACT's mechanism of realigning features before the backbone acts on them: better alignment suppresses the accumulation of physical error over the rollout.

E.6 Learned-map diagnostics

The bounded residual parameterization $\Delta_h = \alpha \tanh(\Delta_{h,\mathrm{raw}})$ yields the componentwise bound $|\Delta_{h,j}| \le \alpha$ and hence $\|\Delta_h\|_2 \le \sqrt{2}\,\alpha$ in two dimensions. We evaluate the learned maps of a fixed SW checkpoint with $\alpha = 0.5$ over a 50-step autoregressive rollout using the diagnostics of Section E.1. Table 13 reports displacement-norm statistics, statistics of $\det J_h$ (Eq. (69)), and the folding rate at rollout steps 0 and 49.

The learned displacements are small relative to the prescribed bound: the global mean and 99th percentile are 0.013 and 0.033, and the maximum norm is 0.266, well below the two-dimensional bound $\sqrt{2} \times 0.5 \approx 0.71$. Across the four blocks the mean Jacobian determinant ranges from 0.977 to 0.993 and the 5th percentile remains positive (0.843–0.886). The global folding rate is 0.14% at rollout step 0 and 0.12% at step 49, with no net increase between the rollout endpoints. These diagnostics support the empirical regularity of ACT's bounded residual maps: coordinate updates remain small and predominantly orientation preserving throughout the evaluated rollout.

Table 13: **Regularity of learned coordinate maps.** Diagnostics for a fixed SW checkpoint over a 50-step rollout. Displacement norms use normalized coordinates with a per-coordinate bound of 0.5. The 99th and 5th percentiles are denoted by p99 and p5. Gray shading identifies the global summary; — denotes an unavailable statistic.

ACT block	Displacement norm			$\det J_h$		Folding rate	
	Mean	p99	Max	Mean	p5	Step 0	Step 49
0	0.0135	0.0267	0.042	0.977	0.886	0.01%	0.03%
1	0.0112	0.0383	0.089	0.988	0.847	0.39%	0.30%
2	0.0111	0.0262	0.096	0.993	0.843	0.17%	0.03%
3	0.0164	0.0720	0.266	0.987	0.861	0.49%	0.13%
Global	0.013	0.033	0.266	0.986	—	0.14%	0.12%

Table 14: **Steady non-advective PDE results.** NRMSE in percent (lower is better). Gray cells denote ACT-augmented models. Relative changes compare +ACT with the same vanilla (VAN) backbone; ↓ and ↑ denote error reduction and increase. FLOWERS and ADRNet are standalone baselines whose errors span the VAN/+ACT columns. The best and second-best results per dataset are **bold** and underlined; — indicates that no paired comparison exists.

Dataset	Model	VAN	+ACT	Relative change
DarcyFlow	CNextU	6.21%	**2.65%**	↓ 57.3%
	FNO	25.03%	3.15%	↓ 87.4%
	Transolver	25.25%	3.90%	↓ 84.6%
	FLOWERS [Muser et al., 2026]	2.75%		—
	ADRNet [Zakariaei et al., 2024]	4.03%		—
Helmholtz	CNextU	**9.05%**	9.39%	↑ 3.8%
	FNO	52.17%	11.69%	↓ 77.6%
	Transolver	37.09%	20.54%	↓ 44.6%
	FLOWERS [Muser et al., 2026]	10.55%		—
	ADRNet [Zakariaei et al., 2024]	17.83%		—
Poisson–Gauss	CNextU	**0.717%**	0.907%	↑ 26.5%
	FNO	10.854%	1.542%	↓ 85.8%
	Transolver	5.375%	1.380%	↓ 74.3%
	FLOWERS [Muser et al., 2026]	0.957%		—
	ADRNet [Zakariaei et al., 2024]	2.590%		—

E.7 Steady non-advective problems

To evaluate ACT beyond advection-dominated dynamics, we consider the stationary DarcyFlow, Helmholtz, and Poisson–Gauss benchmarks adopted from Poseidon [Herde et al., 2024]. Table 14 reports test NRMSE (Eq. (64), single-step since the problems are stationary) for the three backbone families with and without ACT, together with FLOWERS and ADRNet.

Although ACT was motivated by transport, Table 14 shows that it is equally effective on stationary elliptic problems with no advection at all. For the two global operators the improvement is uniform and large: ACT reduces the FNO error by 87.4%, 77.6%, and 85.8% and the Transolver error by 84.6%, 44.6%, and 74.3% on DarcyFlow, Helmholtz, and Poisson–Gauss, respectively. In every case this turns a baseline that is far from competitive (errors of 5–52%) into one that matches or exceeds the specialized models: on DarcyFlow, ACT-FNO (3.15%) and ACT-Transolver (3.90%) both outperform ADRNet (4.03%); on Helmholtz, ACT-FNO (11.69%) is within 1.2 percentage points of FLOWERS (10.55%) and well below ADRNet (17.83%); on Poisson–Gauss, ACT-Transolver (1.380%) and ACT-FNO (1.542%) are within roughly a factor of two of the best model and ahead of ADRNet (2.590%). On DarcyFlow, ACT also improves the convolutional backbone from 6.21% to 2.65%, the lowest error of all evaluated models, ahead of FLOWERS (2.75%). On Helmholtz and Poisson–Gauss, where vanilla CNextU is already the strongest model (9.05% and 0.717%), ACT-CNextU remains the second-best model on Poisson–Gauss (0.907%, ahead of FLOWERS and ADRNet) and within 0.4 percentage points of the best model on Helmholtz (9.39%, ahead of FLOWERS and ADRNet as well). Taken together, ACT yields the best or second-best result on every steady benchmark and never falls behind the standalone warping and advection-aware baselines,

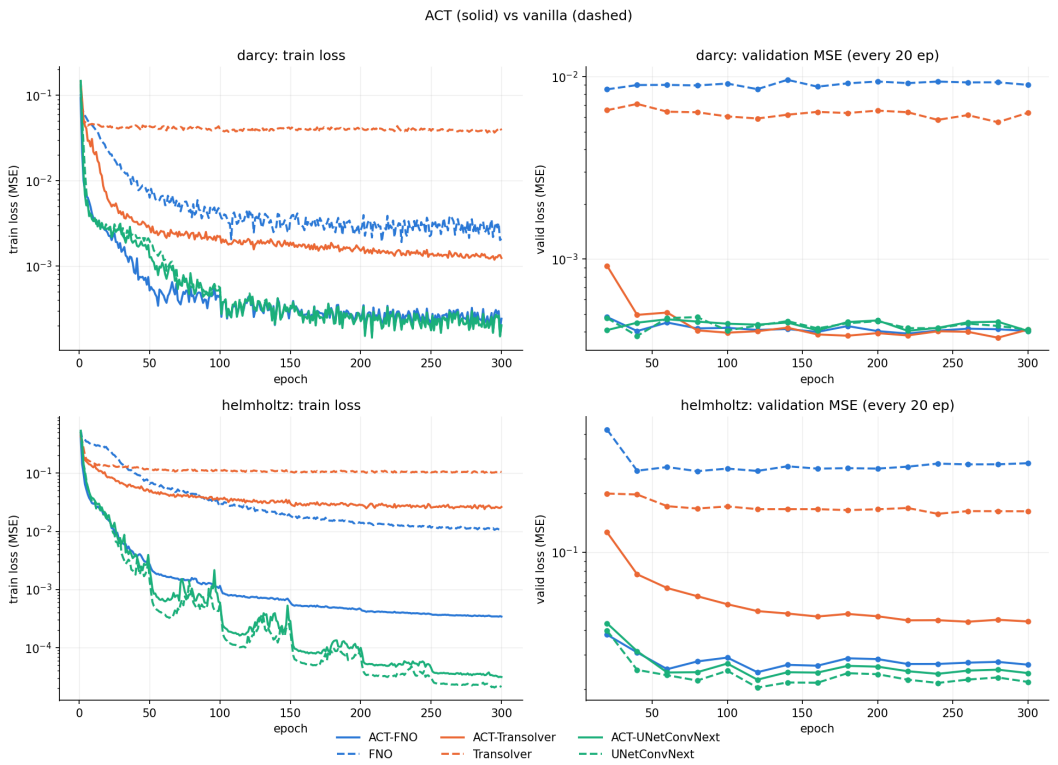

Figure 4: **Training dynamics on the steady benchmarks.** Top: DarcyFlow; bottom: Helmholtz. Left: per-epoch training loss (MSE on normalized targets); right: validation MSE evaluated every 20 epochs. Solid lines are ACT variants and dashed lines the corresponding vanilla backbones; color encodes the backbone (FNO, Transolver, ConvNeXt U-Net). All runs share the same recipe (300 epochs, AdamW with initial learning rate 5×10^{-4} and step decay, batch size 64, two-GPU data parallelism). Both axes are logarithmic.

which supports its applicability well beyond advection-dominated dynamics. The smaller margin for CNextU is expected: a convolutional backbone already has a local receptive field aligned with the grid, so it has less to gain from coordinate realignment than the global spectral and attention operators.

Convergence of all models. Figure 4 shows the training and validation curves of the six backbone/ACT configurations on DarcyFlow and Helmholtz under the shared protocol. In every case the validation error reaches a plateau well before the end of the 300-epoch budget: on DarcyFlow the validation MSE of all six models is flat from roughly epoch 100 onward, and on Helmholtz from roughly epoch 150 onward, with fluctuations between successive validation points that are small compared with the gaps between models. The training loss has likewise saturated, decreasing by less than a factor of two over the final 100 epochs for every run; the residual slow decrease visible for some runs (e.g. vanilla FNO on Helmholtz) is not accompanied by any improvement in validation error. We therefore regard every model as trained to convergence under this protocol, so that the differences in Table 14 reflect the architectures rather than unequal training budgets. We also note that the plateau levels of the vanilla FNO and Transolver on DarcyFlow lie one to two orders of magnitude above those of their ACT counterparts: the gap is established early in training and does not close with additional epochs.

E.8 Positioning against learned spatial and advection-aware models

Spatial transformer networks use an input-conditioned localization network, a grid generator, and differentiable sampling, while deformable convolutions learn offsets for the sampling points of a local convolution. ACT brings adaptive sampling into neural operator blocks through bounded, multi-

Table 15: **One-step comparison with FLOWERS and ADRNet.** Lower NRMSE is better. SW values are scaled by 10^{-4}. Best result in each column is in **bold**; gray rows denote ACT-augmented models.

Model	NS NRMSE ↓	SW NRMSE ($\times 10^{-4}$) ↓	MHD_{64} NRMSE ↓
CNextU	0.0613	2.11	0.1208
ACT-CNextU	0.0496	1.58	0.0963
FNO	0.0222	1.38	0.1540
ACT-FNO	**0.0062**	0.93	0.0909
Transolver	0.3229	3.49	0.2930
ACT-Transolver	0.0135	0.78	**0.0682**
FLOWERS [Muser et al., 2026]	0.0084	**0.56**	0.1047
ADRNet [Zakariaei et al., 2024]	0.0897	1.12	—

head residual coordinate updates. Its one-step improvements with CNextU, FNO, and Transolver (Table 15), together with the rollout improvements for FNO and Transolver (Table 10), show that this operator-level integration transfers across convolutional, spectral, and attention-based architectures.

Table 15 compares one-step NRMSE with FLOWERS and ADRNet on the three time-dependent benchmarks. For readability, SW errors are given in units of 10^{-4}; NS and MHD errors are reported on their original scale.

ACT-augmented backbones take the best result on two of the three benchmarks and are competitive on the third. On NS, ACT-FNO obtains the lowest one-step NRMSE (0.0062), 26% below FLOWERS (0.0084) and an order of magnitude below ADRNet (0.0897). On MHD_{64}, where ADRNet is not applicable, ACT-Transolver (0.0682) and ACT-FNO (0.0909) occupy the first two places, both ahead of FLOWERS (0.1047), and even ACT-CNextU (0.0963) outperforms FLOWERS. On SW, FLOWERS retains the lowest error (0.56×10^{-4}), but ACT brings Transolver and FNO to 0.78×10^{-4} and 0.93×10^{-4}, ahead of ADRNet (1.12×10^{-4}) and within a factor of 1.4–1.7 of FLOWERS, starting from backbones that were 2–6 times worse. These results show that a generic operator backbone equipped with ACT matches or surpasses architectures that were designed around warping or advection from the ground up, while retaining the backbone's own strengths and requiring no architecture-specific redesign.

F Supplementary Visualization Results

In this section, we provide comprehensive visualization results for the remaining benchmark datasets evaluated in our study. Specifically, we present qualitative comparisons between the vanilla baseline architectures (Transolver, FNO, and CNextU) and their ACT-augmented counterparts. The following figures illustrate the model predictions across various physical scenarios, including the scalar vorticity field w in the `Navier_Stokes` dataset, the activator u in the `Diff_Reaction` dataset, the fluid depth h in the `Shallow_Water` dataset, the scalar concentration c in the `Active_Matter` dataset, and the fluid density ρ in the `MHD_64` dataset. These supplementary visual comparisons further corroborate the enhanced spatial representation and fine-grained predictive accuracy introduced by the ACT module across diverse physical dynamics.

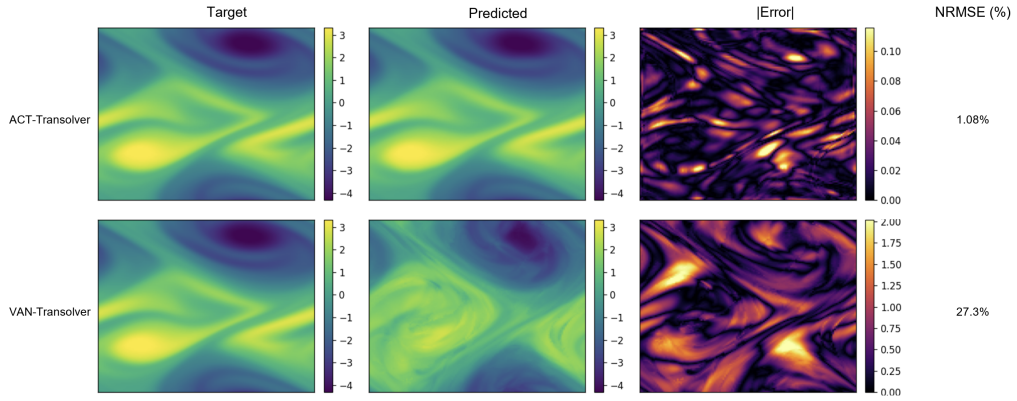

Figure 5: The scalar vorticity field w on the `Navier_Stokes` dataset comparing the vanilla Transolver and ACT-Transolver.

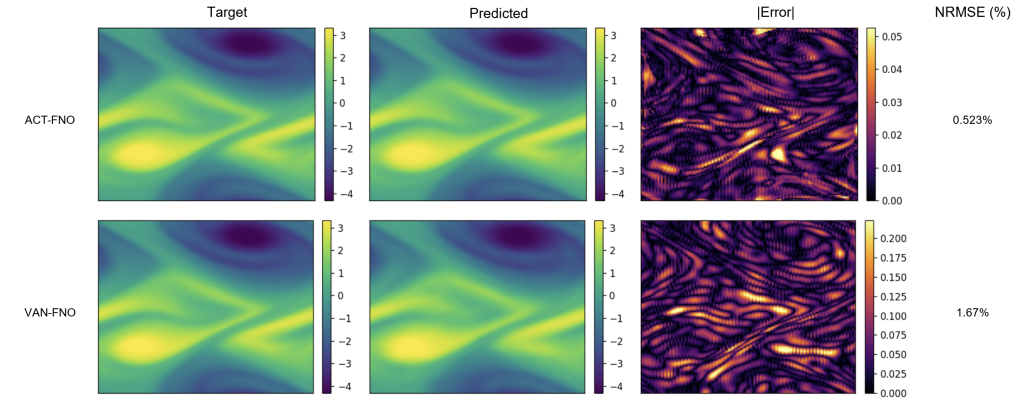

Figure 6: The scalar vorticity field w on the `Navier_Stokes` dataset comparing the vanilla FNO and ACT-FNO.

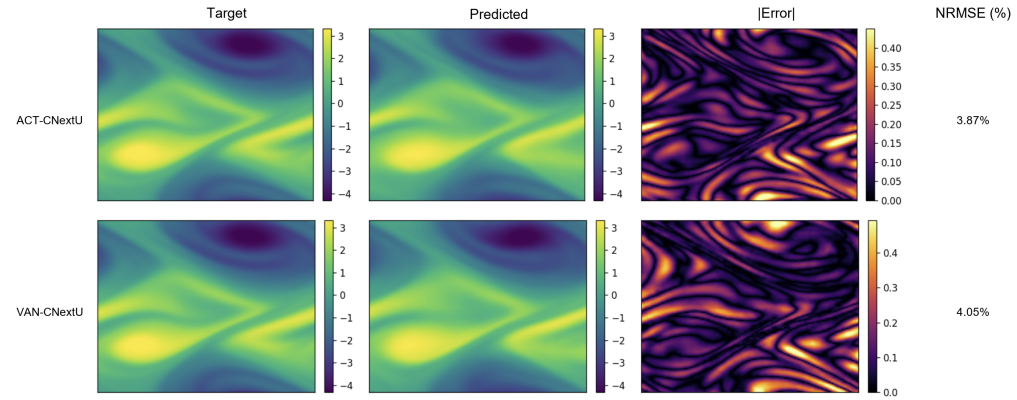

Figure 7: The scalar vorticity field w on the `Navier_Stokes` dataset comparing the vanilla CNextU and ACT-CNextU.

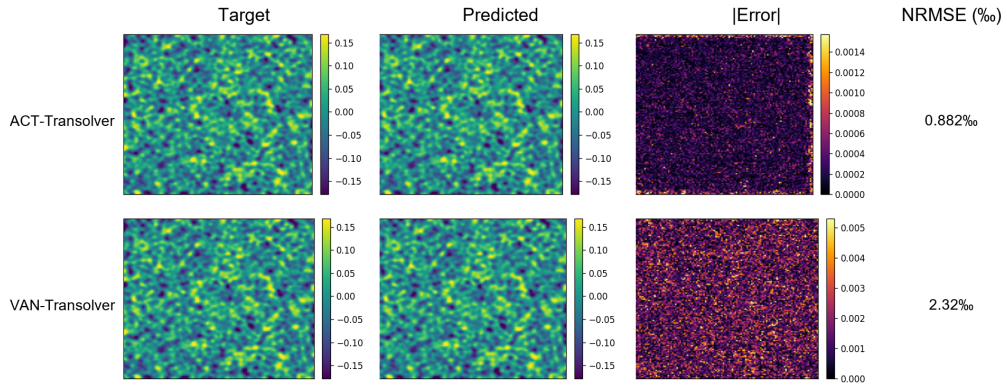

Figure 8: The activator u on the `Diff_Reaction` dataset comparing the vanilla Transolver and ACT-Transolver.

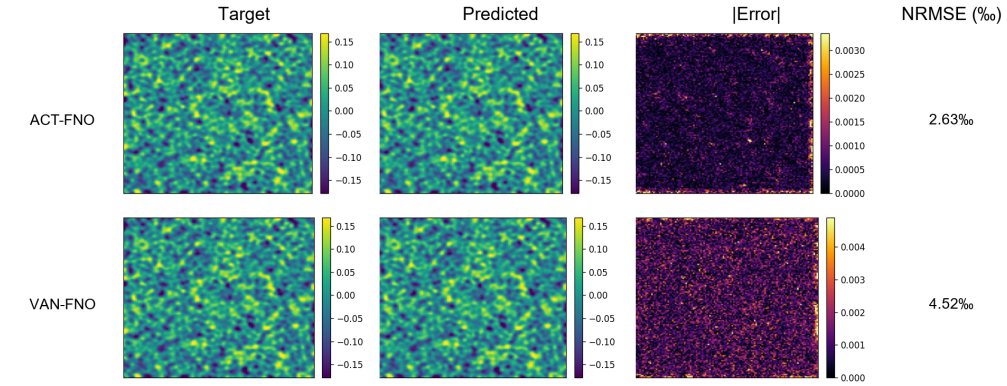

Figure 9: The activator u on the `Diff_Reaction` dataset comparing the vanilla FNO and ACT-FNO.

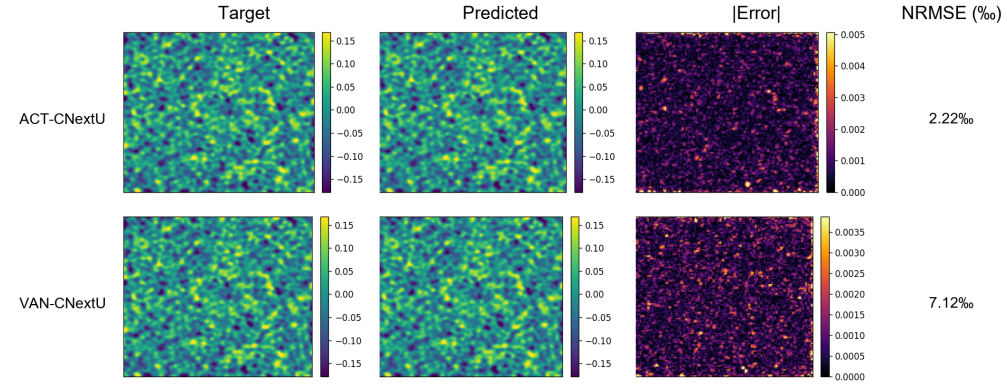

Figure 10: The activator u on the `Diff_Reaction` dataset comparing the vanilla CNextU and ACT-CNextU.

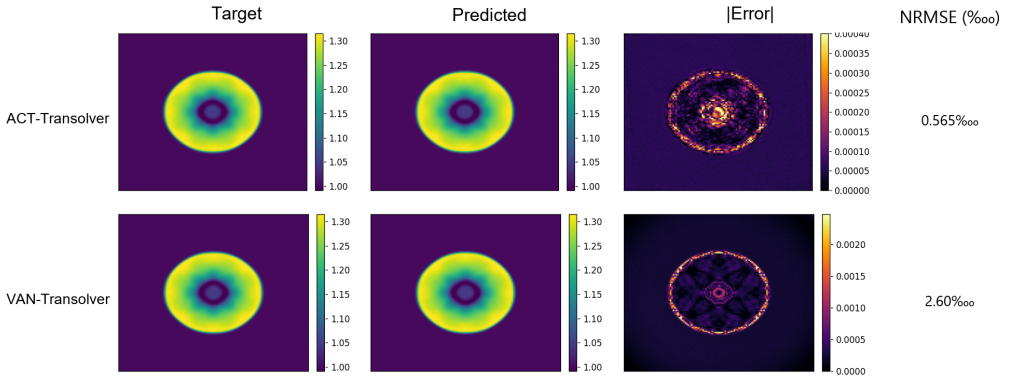

Figure 11: The fluid depth h on the `Shallow_Water` dataset comparing the vanilla Transolver and ACT-Transolver.

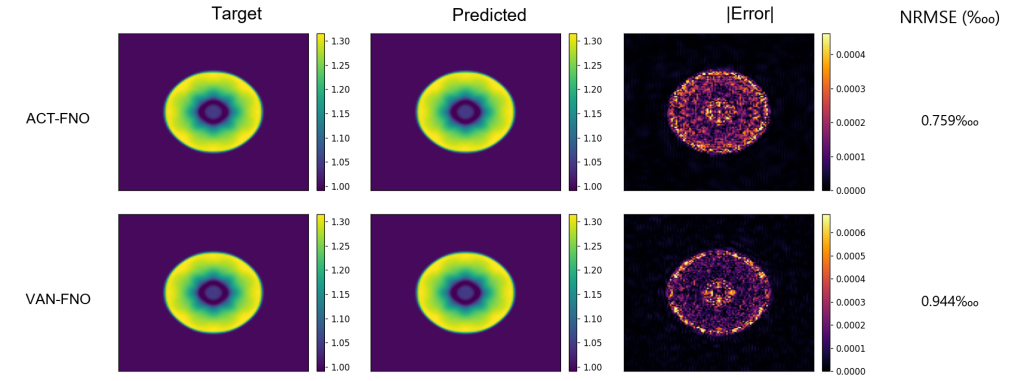

Figure 12: The fluid depth h on the `Shallow_Water` dataset comparing the vanilla FNO and ACT-FNO.

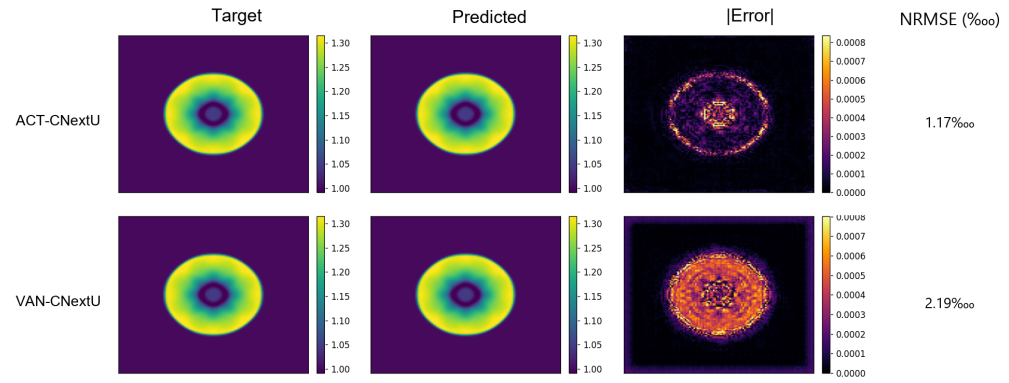

Figure 13: The fluid depth h on the `Shallow_Water` dataset comparing the vanilla CNextU and ACT-CNextU.

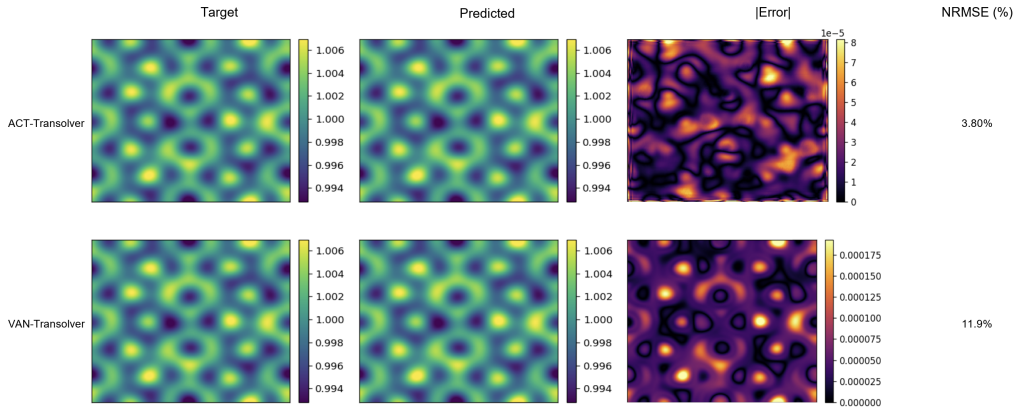

Figure 14: The scalar concentration c on the `Active_Matter` dataset comparing the vanilla Transolver and ACT-Transolver.

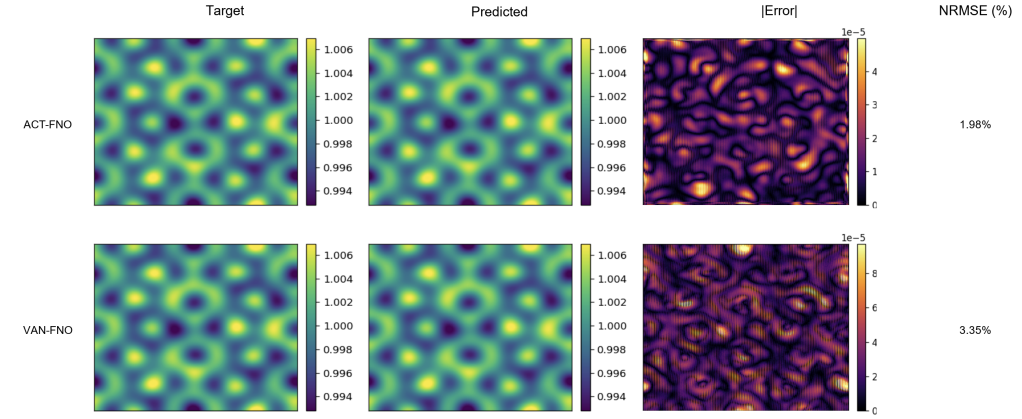

Figure 15: The scalar concentration c on the `Active_Matter` dataset comparing the vanilla FNO and ACT-FNO.

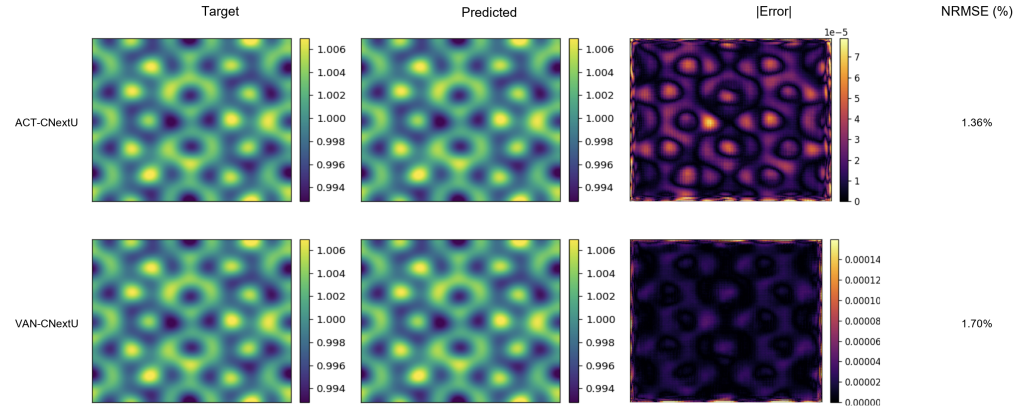

Figure 16: The scalar concentration c on the `Active_Matter` dataset comparing the vanilla CNextU and ACT-CNextU.

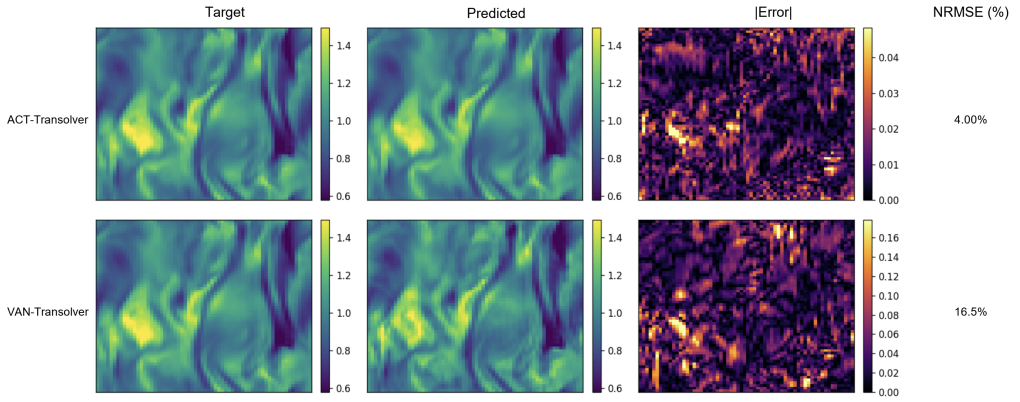

Figure 17: The fluid density ρ on the `MHD_64` dataset comparing the vanilla Transolver and ACT-Transolver.

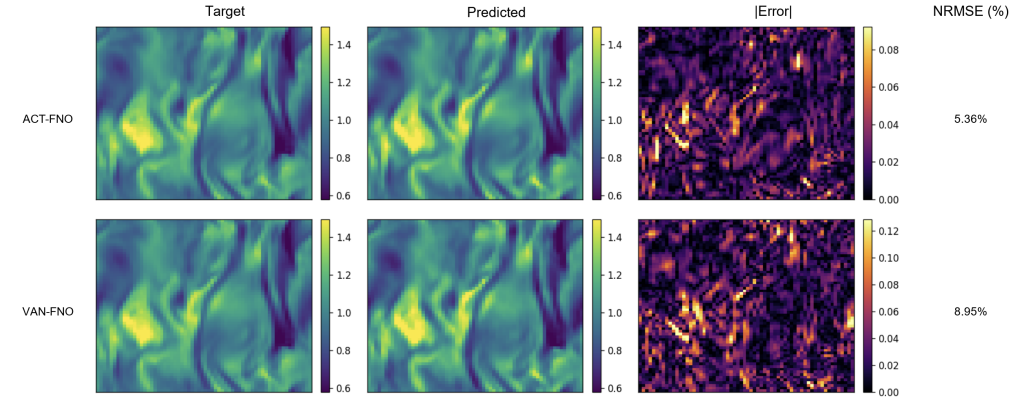

Figure 18: The fluid density ρ on the `MHD_64` dataset comparing the vanilla FNO and ACT-FNO.

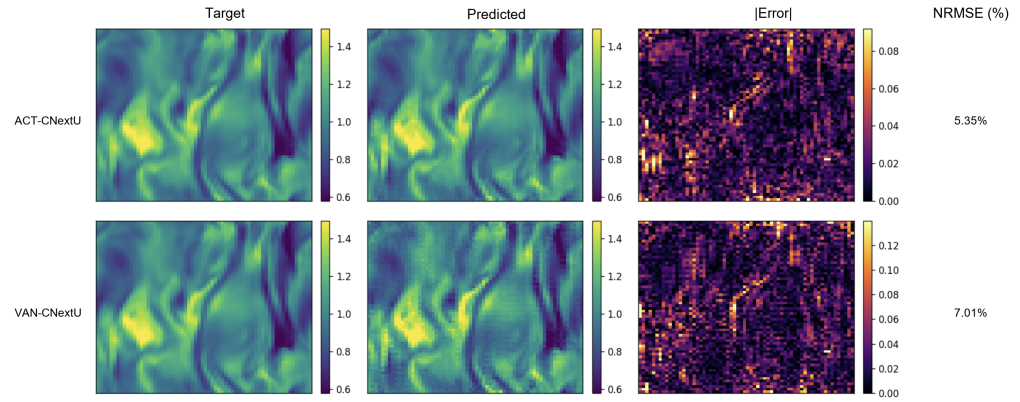

Figure 19: The fluid density ρ on the `MHD_64` dataset comparing the vanilla CNextU and ACT-CNextU.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses limitations in the Conclusion and Limitation section.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not present formal theorems or proofs; instead, it provides conceptual and analytical insights into how coordinate transformations affect operator representations.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper provides detailed descriptions of model architectures, datasets, evaluation metrics, and training protocols sufficient to reproduce the results.

   Guidelines:

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

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

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

Answer: [No]

Justification: While we plan to release the code upon publication, it is not included in the current submission to preserve anonymity. All implementation details are described to support reproducibility.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: Training details, datasets, model configurations, and evaluation protocols are clearly specified in the main text and appendix.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: Due to computational cost, experiments are reported without multiple runs or error bars. However, improvements are consistent across multiple datasets and architectures.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper specifies the hardware setup and training configurations used for experiments (see Appendix).

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics. It involves no human subjects, sensitive data, or high-risk applications.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This work focuses on foundational methodology for PDE learning and does not have immediate societal deployment implications.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The work does not introduce models or datasets with potential misuse risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All datasets and prior methods used are properly cited, and their usage follows standard academic practice.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [No]

    Justification: The paper does not introduce new datasets. Code release is planned but not included in this submission.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The work does not involve human subjects or crowdsourcing.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The work does not involve human subjects.

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were not used as part of the core methodology.

    Guidelines:

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