MOVING-LATENT NEURAL PDE SURROGATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural surrogates offer data-driven alternatives to computationally expensive numerical solvers of partial differential equations (PDEs) by learning to approximate PDE solution operators directly from simulated trajectories. Among various approaches, latent-space surrogates further reduce the cost of long-horizon prediction by evolving compact representations instead of dense physical fields. However, existing approaches typically evolve latent features on fixed spatial or static abstract supports, limiting their ability to adapt where representational capacity is allocated as the solution moves and deforms. We introduce ***Mo**ving-**La**tent Neural **PDE** Surrogates* (MoLa-PDE), where latent features evolve together with their spatial coordinates, allowing a compact latent representation to adapt not only what information is stored but also where it is represented as the PDE dynamics evolve. Making such a moving representation effective requires both modeling interactions on evolving spatial supports and learning how the support itself should deform. To address the former, we develop a geometry-structured continuous operator that explicitly combines feature-dependent interactions with local neighborhood geometry through localized polynomial–Fourier bases. To address the latter, we introduce a structured latent motion mechanism that infers potential fields from global interactions and local differential features, and converts them into support deformation through a Hodge-inspired construction. Experiments across controlled synthetic problems and diverse PDE benchmarks show that the proposed moving-latent representation consistently improves prediction accuracy over fixed-support latent surrogates, while revealing interpretable latent-motion patterns that reflect evolving spatial structures.

1 INTRODUCTION

Partial differential equations (PDEs) form the mathematical backbone of computational models for a wide range of physical and engineering systems, driving predictive modeling in critical applications such as fluid dynamics (Flandoli et al., 2023; Liu et al., 2024), climate forecasting (Verma et al., 2024; Kochkov et al., 2024), and materials discovery (Khatamsaz et al., 2023). However, high-fidelity numerical solvers for PDEs are often computationally expensive, particularly for high-resolution simulations, long temporal horizons, or repeated evaluations (Lapidus & Pinder, 1999; Müller & Scheichl, 2014). Neural PDE surrogates have emerged as an efficient alternative for learning solution dynamics directly from simulated or observed trajectories, with recent advances spanning physics-informed learning (Raissi et al., 2019), neural operators (Li et al., 2023c; Kovachki et al., 2023), neural simulators (Maddu et al., 2023; Wei et al., 2026), and deep generative models (Shysheya et al., 2024; Huang et al., 2024).

Despite their efficiency over conventional solvers, many neural PDE surrogates still evolve solution states on dense spatial discretizations, making temporal prediction increasingly costly as the spatial resolution grows Fortunato et al. (2022); Brandstetter et al. (2022); Lippe et al. (2023). Latent-space PDE surrogates (Wang & Wang, 2024; Serrano et al., 2024; Hagnberger et al., 2025) alleviate this bottleneck by compressing the physical field into a compact set of latent states and performing temporal evolution in the latent space. This substantially reduces the cost of recurrent dynamics while preserving the ability to reconstruct the solution in the physical domain.

However, existing latent formulations typically organize latent states through abstract indices or fixed spatial supports (Wang & Wang, 2024; Serrano et al., 2024; Chen & Wu, 2024; Hagnberger et al., 2025). Consequently, temporal evolution mainly updates latent features, while the geometry

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

supporting these features remains static. Such a fixed latent geometry can be restrictive for PDE systems dominated by transport, advection, or deformation, as evolving structures must be represented through feature changes alone. It also limits the model's ability to adaptively redistribute spatial resolution toward regions with sharp gradients or localized dynamics.

Motivated by these limitations, we introduce ***Mo**ving-**La**tent Neural **PDE** Surrogates* (MoLa-PDE), which explicitly include the spatial coordinates in latent states and evolve them jointly with the latent features. This allows a compact latent state to adapt both what information is represented and where capacity is allocated as the PDE dynamics evolve. To make such a moving latent space effective, we develop a geometry-structured continuous operator that explicitly combines feature-dependent interactions with neighborhood geometry and a structured latent motion mechanism that infers potential fields from global context and local messages to generate coherent support deformation. Integrated into an encoder–evolver–decoder architecture, these components enable latent features and spatial geometry to coevolve throughout the autoregressive rollout, enabling the model to better track evolving spatial structures while producing accurate predictions. Overall, our main contributions are threefold:

- We introduce MoLa-PDE, a moving-latent neural PDE surrogates, where latent features and spatial coordinates evolve jointly, allowing the latent state to adapt both what and where information is represented.
- We develop a geometry-structured continuous operator for evolving spatial interactions and a potential-based motion mechanism for structured support deformation, enabling effective moving latent dynamics.
- We conduct extensive experiments on synthetic problems and PDE benchmarks. The results demonstrate consistent benefits of latent motion over the corresponding fixed-support models.

2 Background

In this section, we review the standard time-dependent PDEs and neural PDE surrogate models, and introduce a typical latent-space formulation for neural PDE surrogates. A broader discussion of related neural PDE architectures and the positioning of our method is provided in Appendix A.

2.1 PDE and neural PDE surrogates

We consider time-dependent PDEs formulated by $\partial_t u(t,x) = \mathcal{F}\big(u, \nabla u, \nabla^2 u, ..., x, t, \lambda\big)$,where $x \in \Omega \subset \mathbb{R}^d$ is the spatial coordinate, $t \in [0,T]$ indexes the temporal domain, $u(t,x)$ denotes the solution field, and $\mathcal{F}$ is a differential operator governing the underlying dynamics. The dynamics are determined together with appropriate initial value $u(0,x) = u_0(x)$ and boundary condition $u(t,x)\big|_{\partial\Omega} = \mathcal{B}(t,x)$.

Neural PDE surrogates provide an alternative by learning the evolution directly from trajectories generated by computationally expensive numerical simulations (Pestourie et al., 2023; Kovachki et al., 2023; Huang et al., 2025). Given a set of training trajectories sampled at discrete time steps and spatial locations, our goal is to learn a neural surrogate model that approximates the solution operator $\mathcal{G}_\theta : (u_0, \mathcal{F}) \mapsto \{u(t,\cdot)\}_{t\in[0,T]}$, where θ denotes the neural surrogate parameters to be learned. In this paper, we consider a discrete-time, continuous-space setting, where the solution is evaluated at a finite sequence of time steps $\{t_k\}_{k=0}^K$ and can be queried at arbitrary spatial coordinates $x \in \Omega$.

2.2 Latent-space neural PDE surrogates

Instead of directly evolving the high-dimensional solution field $u(t,x)$ on dense spatial grids with size N, latent-space methods first compress the dense solution into a compact representation and then model its temporal evolution in a lower-dimensional latent space (Wang & Wang, 2024; Serrano et al., 2024; Hagnberger et al., 2025).

Concretely, given an initial point cloud field $\mathbf{U}_0 \in \mathbb{R}^{N\times C_{in}}$, an encoder, typically parameterized by a neural network such as CNNs or GNNs, maps the field to a collection of latent features by $\mathbf{Z}_0 = \mathcal{E}_{\theta_\mathcal{E}}\big(\mathbf{U}_0, \mathbf{X}\big)$, where $Z_0 \in \mathbb{R}^{M\times C_{latent}}$ denotes M latent features, each of which could be regarded as a compact representation associated with an anchor in the latent domain. Typically,

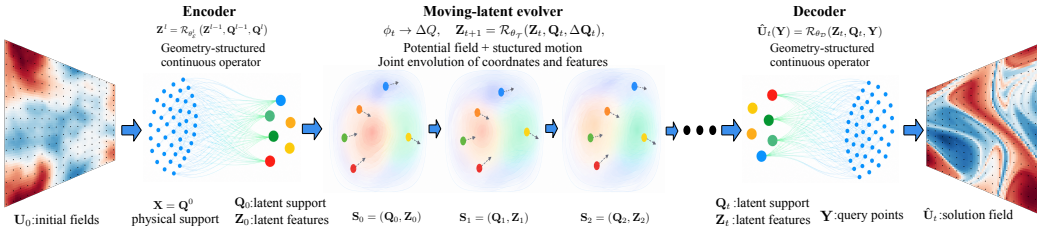

Figure 1: MoLa-PDE Surrogate with an encoder–evolver–decoder architecture. The encoder maps the initial physical field onto a compact spatial latent state using the geometry-structured continuous operator. The moving-latent evolver jointly updates coordinates and features through structured motion and geometry-aware feature evolution. The decoder maps the evolved latent state back to arbitrary physical query locations.

$M \ll N$ and $MC < N$, substantially reducing the computational cost of temporal evolution in the latent space. The latent features are then propagated across discrete time steps through a learned transition model, $\mathbf{Z}_{t+1} = \mathcal{T}_{\theta_\mathcal{T}}(\mathbf{Z}_t)$, which models the temporal dynamics entirely in the latent space. At the desired time steps, a decoder maps the evolved latent features back to the physical domain to reconstruct the solution field by $\hat{\mathbf{U}}_t = \mathcal{D}_{\theta_\mathcal{D}}(\mathbf{Z}_t, \mathbf{Y})$.

While latent-space surrogates substantially reduce the cost of temporal evolution, a common formulation defines latent representations either over abstract indices without explicit spatial meaning or over fixed coordinates. Translation models only process the latent features, leaving the spatial support unchanged. This can be restrictive for transportation, deformation, and localized moving structures, and also limits adaptive allocation of support to sharp or information-rich regions.

3 Moving-Latent Neural PDE Surrogates

Unlike previous latent-space surrogates in Sec. 2.2, we formulate support coordinates as dynamic variables rather than static grids, jointly modeling feature dynamics and evolving latent spatial geometries. This allows a fixed number of latent states to adapt both what they represent and where they allocate representational capacity in the physical domain. To achieve this, we adopt an encoder–evolver–decoder architecture, as shown in Fig. 1. The encoder and decoder are built upon a geometry-structured continuous operator, which transfers information between spatial supports through continuous functions of their local neighborhood geometry. The evolver advances both locations and features of the spatial latent state $\mathbf{S}_t = (\mathbf{Q}_t, \mathbf{Z}_t)$, coupling a structured deformation of the latent support $\mathbf{Q}_t$ and dynamic feature $\mathbf{Z}_t$ evolution on the resulting geometry.

In this section, we first introduce the shared geometry-structured continuous operator modules and the key structured latent motion mechanism, then describe how we assemble them into the complete encoder–evolver–decoder architecture.

3.1 Geometry-Structured Continuous Operator

Moving latent representations make the support coordinates themselves part of the dynamical state. Therefore, changes in the latent geometry should directly affect how information is transferred between latent states. A common approach is to model continuous spatial interactions with a generic neural network over absolute or relative coordinates, possibly together with feature-dependent modulation (Wang et al., 2018; Wei et al., 2023; Kim & Park, 2023; Hagnberger et al., 2025). While expressive, such a formulation entangles geometric and feature dependence inside a learned nonlinear mapping. This is less desirable in our setting, where the latent support is explicitly evolved, and the geometry is intended to play a direct role in the spatial operator.

We consider a generic source–target interaction, where $\mathbf{P} = \{P_j\}_{j=1}^{N_s}$ denotes a source support carrying features $\mathbf{Z} = \{Z_j\}_{j=1}^{N_s}$, and $\mathbf{Q} = \{Q_i\}_{i=1}^{N_t}$ denotes a target support at which new features are evaluated. Under this formulation, we therefore explicitly factorize features and geometry: source features determine the coefficients of a continuous local kernel, while the source–target geometry

determines its evaluation through a prescribed geometric basis. Consequently, moving the latent support directly changes the operator through the local neighborhood geometry, rather than affecting it only through an unconstrained neural parameterization. For each source–target pair, we construct the pairwise value as

$$v_{ijc} = \sum_{p=1}^{N_P} c_{jcp}(Z_j)\, \beta_p(\xi_{ij}), \qquad \xi_{ij} = \frac{\Delta_\Omega(Q_i, P_j)}{\ell}. \tag{1}$$

Here, $\{\beta_p\}_{p=1}^{P}$ are prescribed basis functions of the normalized local neighborhood geometry, $c_{jcp}(Z_j)$ are coefficients predicted from the source feature Z_j by a learnable network with c indexing the output channel, ℓ is a characteristic support spacing defined by $N_s^{-1/d}$, and Δ_Ω denotes the domain- and boundary-aware relative displacement.

Localized polynomial–Fourier basis. We construct the relative basis set $\mathcal{B}_d(\xi)$ by combining low-order polynomial and Fourier components. The polynomial terms provide a compact representation of smooth local spatial variation, while the Fourier terms complement them with oscillatory and directional patterns at different frequencies (Tancik et al., 2020). To preserve locality, we modulate all components with Gaussian localization factors, so their contributions decay with the relative source–target distance. For a d-dimensional domain, the basis is formulated by

$$\begin{aligned}\mathcal{B}_d(\xi) =& \{e_p(\xi)\xi_a\}_{a=1}^{d} \cup \{e_p(\xi)\xi_a\xi_b\}_{1\le a\le b\le d} \\ &\cup \left\{e_\kappa(\xi)\sin(2\pi\kappa\, u^\top\xi),\ e_\kappa(\xi)\left[\cos(2\pi\kappa\, u^\top\xi) - 1\right]\right\}_{\kappa\in\mathcal{K},\ u\in\mathcal{U}_d}.\end{aligned} \tag{2}$$

where $\mathcal{K}$ denotes a set of frequency scales and $\mathcal{U}_d$ a set of spatial directions. We localize components using $e_p(\xi) = \exp(-\kappa_p^2\|\xi\|_2^2/2)$ and $e_\kappa(\xi) = \exp(-\kappa^2\|\xi\|_2^2/2)$, where κ_p controls the localization of the polynomial terms, while κ jointly determines the frequency and localization of each Fourier component. Both the sine and centered-cosine components vanish at zero displacement. These terms and corresponding parameters are predefined and are fixed. Specific frequency scales and directions for different spatial dimensions are provided in Appendix B.1.

Multihead feature-aware aggregation. To capture expressive spatial interactions, we instantiate the operator with multiple heads, each learning its own feature-conditioned pairwise values and source aggregation pattern. For head h, the contribution of source j to target i is weighted by

$$\alpha_{ijh} = \mathrm{softmax}_{j\in\mathcal{N}(i)}\left(a_{ijh} - \lambda_h\|\xi_{ij}\|_2^2\right), \qquad \lambda_h = \mathrm{softplus}(\rho_h) + \epsilon, \tag{3}$$

where a_{ijh} is a feature-dependent relevance score, $\lambda_h > 0$ controls the spatial range of head h, and $\mathcal{N}(i)$ denotes the candidate source set. For cross-support operations, where the target has no associated feature, a_{ijh} is predicted from the source feature alone; for interactions of same supports, it is computed using a query–key similarity (Vaswani et al., 2017). The head-wise outputs are then calculated by

$$\mathcal{R}_\theta(\mathbf{Z}, \mathbf{P}, \mathbf{Q})_i = W_o\, \mathrm{Concat}_h\Big(\sum_{j\in\mathcal{N}(i)} \alpha_{ijh}\mathbf{v}_{ijh}\Big) \tag{4}$$

as the final output of our operator. Here, W_o denotes a learnable output projection that mixes the aggregated head-wise features. Each head has independent basis coefficients and aggregation parameters, which allows the operator to capture complementary spatial interaction patterns, with different heads learning distinct feature dependencies and effective spatial ranges.

3.2 Structured Latent Motion

The motion module in the evolver predicts a deformation of the latent support according to the current latent dynamics. A straightforward approach is to predict an independent displacement vector at each latent location using a neural network. However, such pointwise predictions do not explicitly couple the motion of neighboring latent states, making it harder to preserve local coherence or capture collective deformation patterns. Instead of this, we construct the motion through latent potential fields. This introduces an explicit spatial structure into the motion field, as the displacement is derived from the spatial variation of neighboring potential values under the current latent geometry. Since all quantities in the following are evaluated at the same current time step t, we omit the time index.

Latent Potential Construction. Given the current latent state set $\mathbf{S} = \{(Q_i, Z_i)\}_{i=1}^{M}$, we predict the latent potentials using the current latent feature together with global and local context, $\Phi_i =$

$f_{\theta_p}(Z_i, G_i, L_i)$, where Z_i, G_i, and L_i denote the current latent feature, global and local context, respectively. These complementary contexts allow the motion predictor to account for both global interactions and spatial variation around each latent location.

Specifically, the global context is obtained by applying a same-support instance of the geometry-structured operator, $\mathbf{G} = \mathcal{R}_{\theta_g}(\mathbf{Z}, \mathbf{Q}, \mathbf{Q})$, which allows each latent state to incorporate information from the entire current latent representation and provides support-wide context for coordinating the deformation of different latent locations.

The local context L_i is designed to capture two complementary aspects of the neighborhood relations: how strongly the latent features differ from their surroundings, and how these variations are organized across spatial directions. Accordingly, we use a neighborhood contrast feature C_i to summarize how the current latent feature differs from its surroundings, and a differential feature D_i^Z to characterize how these differences vary across spatial directions. Given the relative geometry ξ_{ij}, we define the neighborhood weights as $w_{ij} = \mathrm{softmax}_{j\neq i}(-\|\xi_{ij}\|_2^2)$. We first construct the neighborhood contrast feature as $C_i = g_i \odot \sum_{j\neq i} w_{ij}(Z_j - Z_i)$, where g_i is a feature-dependent gate. This summarizes how the current latent feature differs from its neighboring states while emphasizing spatially closer interactions. To further capture the directional structure of these local differences, we estimate a feature Jacobian through the weighted local fit

$$J_i^Z = \arg\min_J \sum_{j\neq i} w_{ij} \left\|(Z_j - Z_i) - J\xi_{ij}\right\|_2^2, \tag{5}$$

where $J_i^Z \in \mathbb{R}^{C\times d}$ characterizes the directional variation of the latent features around Q_i. In practice, this weighted least-squares problem admits the closed-form solution $J_i^Z = [\sum_{j\neq i} w_{ij}(Z_j - Z_i)\xi_{ij}^\top][\sum_{j\neq i} w_{ij}\xi_{ij}\xi_{ij}^\top]^{-1}$. Since d is the spatial dimension, the computation only involves a small $d \times d$ linear system at each latent location. We further project the feature Jacobian into a compact differential feature, $D_i^Z = W_J\,\mathrm{flatten}(J_i^Z)$, where W_J is a learnable linear projection. Together, V_i captures the magnitude of local feature differences, while D_i^Z retains their directional dependence on the latent geometry. These two components are concatenated to form the local context, $L_i = [V_i, D_i^Z]$. Using these features, we could predict potential values at each latent location by an MLP $\Phi_i = f_{\theta_p}(Z_i, G_i, L_i)$.

Potential Induced Motion. A single scalar potential only represents gradient-type deformation. To further capture rotational motion, we parameterize the local velocity using a scalar potential ϕ and an antisymmetric potential field A, following a Hodge-inspired decomposition. Accordingly, the potential predictor outputs $\Phi_i = [\phi_i, \{\psi_{ab,i}\}_{1\leq a<b\leq d}]$, where the scalar pair potentials ψ_{ab} parameterize the independent components of the antisymmetric field A, with $A_{ab} = \psi_{ab}$ and $A_{ba} = -\psi_{ab}$. For a d-dimensional domain, this introduces $d(d-1)/2$ pair potentials, each associated with a coordinate plane and modeling a corresponding rotational component of the local motion.

Since the potentials are predicted only at the latent locations, we estimate their spatial derivatives directly on the current support. For each potential component $\chi \in \Phi$, we also use the same weighted local fitting scheme, $\widehat{\nabla\chi_i} = [\sum_{j\neq i} w_{ij}(\chi_j - \chi_i)\xi_{ij}^\top][\sum_{j\neq i} w_{ij}\xi_{ij}\xi_{ij}^\top]^{-1}$. The estimated derivatives are then combined according to the Hodge-inspired parameterization (Bhatia et al., 2012) to form the local velocity, $v_{i,a}^{\mathrm{loc}} = \partial_a\phi_i + \sum_{b=1}^d \partial_b A_{ab,i}$. The scalar-potential term provides the gradient component of the motion, whereas the antisymmetric term introduces rotational components across the corresponding coordinate planes. We then convert the local velocity into a bounded local displacement. To additionally represent a coherent translation of the entire latent support, we predict a state-dependent translation vector shared by all latent locations. The resulting motion is

$$\Delta Q_{t,i} = \Delta Q_{t,i}^{\mathrm{loc}} + \Delta Q_t^{\mathrm{trans}}, \qquad Q_{t+1,i} = \Pi_\Omega\left(Q_{t,i} + \Delta Q_{t,i}\right), \tag{6}$$

where Π_Ω applies the corresponding boundary treatment.

3.3 Encoder–Evolver–Decoder Architecture

With the geometry-structured operator and latent motion mechanism established, we now assemble them into the complete moving latent surrogate. The three components share the same spatial operator formulation but use independent parameters for their respective roles.

Encoder. Given an initial field $\mathbf{U}_0$ observed at coordinates $\mathbf{X}$, the encoder progressively maps the physical samples onto a compact latent support. Each stage introduces a lower-dimensional target support and applies the geometry-structured operator in Sec. 3.1 to transfer features from the current source support to the target support. At encoder stage l, $\mathbf{Z}^l = \mathcal{R}_{\theta^l_{\mathcal{E}}}\left(\mathbf{Z}^{l-1}, \mathbf{Q}^{l-1}, \mathbf{Q}^l\right)$. Here, $\mathbf{Q}^0$ is the input position, $\mathbf{Z}^0$ denotes the lifted physical features, and $\mathbf{Q}^l$ is defined on a uniform layout. Repeating the operation over multiple resolutions produces the initial spatial latent state $\mathbf{S}_0 = (\mathbf{Q}_0, \mathbf{Z}_0)$ with substantially fewer spatial states than the physical discretization.

Evolver. Starting from $\mathbf{S}_t = (\mathbf{Q}_t, \mathbf{Z}_t)$, the motion module in Sec. 3.2 updates the latent support to $\mathbf{Q}_{t+1}$. Feature evolution then combines two complementary uses of the geometry-structured operator,

$$\mathbf{Z}_t^{\text{cross}} = \mathcal{R}_{\theta_{\text{cross}}}(\mathbf{Z}_t, \mathbf{Q}_t, \mathbf{Q}_{t+1}), \qquad \mathbf{Z}_{t+1} = \mathcal{R}_{\theta_{\text{same}}}(\mathbf{Z}_t^{\text{cross}}, \mathbf{Q}_{t+1}, \mathbf{Q}_{t+1}), \tag{7}$$

The first instance transfers information from the old support $\mathbf{Q}_t$ to the moved support $\mathbf{Q}_{t+1}$, while the second models interactions among latent states on the updated support. Both follow the geometry-structured operator in Sec. 3.1, but use independent parameters and different feature-routing mechanisms. With the motion update, one recurrent step therefore evolves the spatial latent state as $(\mathbf{Q}_t, \mathbf{Z}_t) \to (\mathbf{Q}_{t+1}, \mathbf{Z}_{t+1})$, allowing latent geometry and features to coevolve throughout the rollout.

Decoder. At any time step, the decoder maps the moving latent representation back to the physical domain by applying the geometry-structured operator from the latent support $\mathbf{Q}_t$ to an arbitrary query set $\mathbf{Y}$ by $\hat{\mathbf{U}}_t(\mathbf{Y}) = \mathcal{R}_{\theta_{\mathcal{D}}}(\mathbf{Z}_t, \mathbf{Q}_t, \mathbf{Y})$. Since the decoder is defined through source–target geometry rather than a fixed grid, the evolved solution can be queried continuously at arbitrary spatial locations.

3.4 Training

We train the complete model end-to-end through autoregressive rollout. Given an initial field $\mathbf{U}_0$, the encoder encodes the latent state, the evolver recurrently advances both its features and spatial support, and the decoder reconstructs the physical field at each prediction step. We minimize the relative prediction error $\mathcal{L}(\theta) = \frac{1}{T}\sum_{t=0}^{T} \frac{\|\hat{\mathbf{U}}_t - \mathbf{U}_t\|_2}{\|\mathbf{U}_t\|_2 + \epsilon}$. The training follows a curriculum strategy where rollout length T is gradually increased over training epochs (Bengio et al., 2009; Hagnberger et al., 2025). This allows the model to progressively learn longer-term autoregressive dynamics. Additionally, we apply small perturbations to the latent support coordinates $\mathbf{Q}$, encouraging robustness to nearby spatial configurations and reducing over-specialization to a single latent-support trajectory. Additional training settings, implementation details, and hyperparameter choices are provided in Appendix B.3.

4 Experiments

Our experiments are designed to perform the following evaluations: **(1)** Can evolving latent supports effectively model PDE dynamics with sharp spatial discontinuities and rapid localized dynamics? **(2)** How does the complete MoLa-PDE perform on standard PDE benchmarks compared with existing physical- and latent-space surrogates? **(3)** How do the key architectural components and design choices affect prediction performance? **(4)** How efficient are the proposed MoLa-PDE in terms of memory usage and inference speed?

4.1 Experimental setup

We evaluate our method on both controlled synthetic problems and standard PDE benchmarks. The controlled problems are designed to isolate when moving latent supports are most useful, while the benchmark experiments evaluate predictive performance on more complex PDE dynamics.

Controlled synthetic problems. We construct three families of two-dimensional synthetic problems to isolate different roles of moving latent supports: *Smooth Transport* (***T***), *Sharp Interface Propagation* (***S***), and *Sharp Transport* (***ST***). ***T*** follows the advection equation $\partial_t u + \mathbf{v} \cdot \nabla u = 0$, with initial conditions generated as mixtures of localized Gaussian blobs and transported by a spatially varying shear flow. This setting is designed to test whether moving latent supports can follow smooth but spatially localized information as it is advected and deformed. ***S*** follows the level-set equation $\partial_t \phi + c(\mathbf{x}, t)|\nabla \phi| = 0$. The sharp initial interfaces are obtained by thresholding smooth Fourier fields, converting the resulting regions to signed-distance functions, and applying a sharp transition profile. The spatially and temporally varying normal speed produces nonuniform interface deformation, allowing us to test adaptive representation around evolving sharp structures without advection. ***ST***

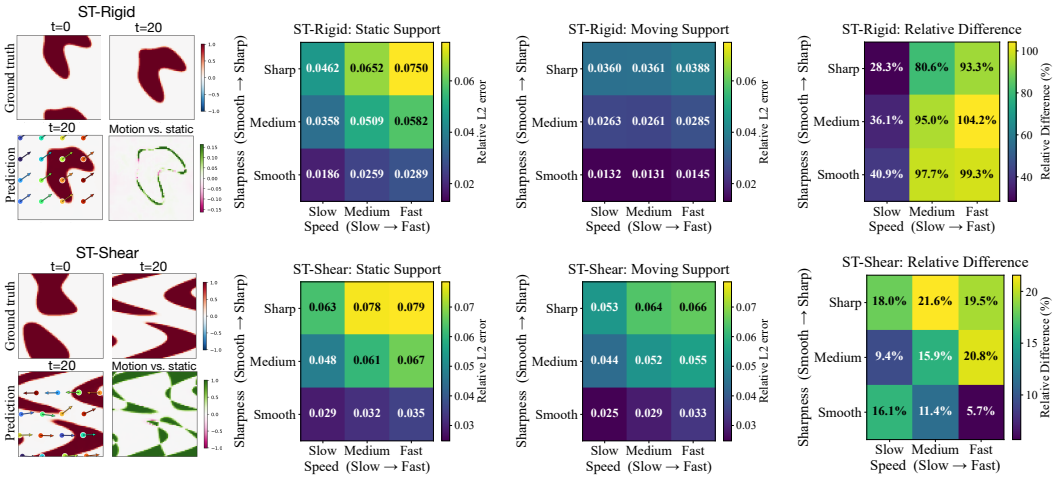

Figure 2: Controlled sharp-transport (***ST***) experiments under rigid and shear dynamics. **Left:** Example trajectories and learned latent-support motion for *ST-Rigid* and *ST-Shear*, illustrating coherent global translation and spatially varying shear motion, respectively. **Right:** Relative L_2 errors of static and moving latent supports, and the relative improvement from latent motion, across different interface sharpness and transport-strength settings.

follows $\partial_t u + \mathbf{v} \cdot \nabla u = 0$, but uses the same class of sharp initial interfaces as ***S***. We consider both a rigid-translation variant, which isolates global relocation, and a shear-transport variant, which induces spatially varying displacements and therefore requires nonuniform latent motion. We further vary interface sharpness and transport strength in a controlled 3×3 grid. All problems use the same spatial resolution and rollout horizon. The model observes only the initial field and spatial coordinates, without access to the underlying velocity field, interface location, or other parameters for data simulation. Detailed initial-condition construction, velocity fields, and parameter values are provided in Appendix C.1.

PDE Benchmarks. We further evaluate our method on a diverse set of one- and two-dimensional PDE benchmarks. From PDEBench(Takamoto et al., 2022), we consider 1D Burgers, Advection, and Reaction–Diffusion equations, and 2D Shallow Water and Navier–Stokes dynamics. We additionally include flow around a cylinder (Pfaff et al., 2021) on an irregular mesh to evaluate the model beyond regular-grid domains and under more complex spatial geometry. These benchmarks cover transport, nonlinear dynamics, diffusion–reaction processes, wave propagation, vortical flow, and irregular spatial discretizations. Dataset-specific settings are provided in Appendix C.2.

Baselines. We compare against a broad set of neural PDE surrogates, including FNO (Li et al., 2023c), F-FNO (Tran et al., 2023), OFormer (Li et al., 2023a), PIT (Chen & Wu, 2024), LNO (Wang & Wang, 2024), AROMA (Serrano et al., 2024), and CALM-PDE (Hagnberger et al., 2025). These baselines cover both physical-space neural operators and latent-space surrogate models. To directly assess the contribution of latent motion, we additionally include *MoLa-PDE (Static)*, which uses the same encoder–evolver–decoder architecture and latent state size as our full model but keeps the latent support fixed throughout the rollout. *MoLa-PDE* denotes our proposed full model with jointly evolving latent features and coordinates. Implementation details and baseline configurations are provided in Appendix D. We evaluate prediction using the relative L_2 error over the autoregressive rollout, averaged across test trajectories.

4.2 Results

We report the main results here, with additional results and latent motion visualizations in Appendix E.

Understanding under Controlled Dynamics. Moving latent supports consistently improve prediction accuracy across the controlled synthetic problems. The complete numerical results for ***T***, ***S***, and all ***ST*** variants are reported in Appendix Table E1. On Smooth Transport (***T***), the relative ℓ_2 error decreases from 0.0232 with static supports to 0.0209 with moving supports, corresponding to an 11.0% relative difference. On Sharp Interface Propagation (***S***), moving supports reduce the error from 0.121 to 0.0986, corresponding to a 22.7% relative difference. These results show that

Table 1: Comparison of relative L_2 error ($\times 10^{-2}$) across PDE benchmarks. Percentages in parentheses denote the relative error difference with respect to MoLa-PDE.

Model	1D Burgers	1D Diffusion–Reaction	1D Advection	2D Navier–Stokes	2D Shallow Water	2D Cylinder
FNO	3.58 (+106%)	0.453 (+178%)	0.938 (+116%)	9.12 (+0.885%)	0.254 (+28.3%)	–
F-FNO	3.62 (+108%)	0.210 (+28.8%)	1.13 (+160%)	<u>8.44</u> (−6.64%)	0.331 (+67.2%)	14.9 (+15.5%)
OFormer	5.75 (+230%)	0.296 (+81.6%)	<u>0.811</u> (+86.9%)	19.4 (+114%)	0.405 (+105%)	22.6 (+75.2%)
PIT	12.1 (+595%)	0.458 (+181%)	4.32 (+895%)	16.3 (+80.6%)	0.265 (+33.8%)	14.0 (+8.53%)
LNO	3.09 (+77.6%)	0.244 (+49.7%)	2.47 (+469%)	**7.89** (−12.7%)	0.267 (+34.8%)	16.5 (+27.9%)
AROMA	9.37 (+439%)	0.280 (+71.8%)	2.57 (+592%)	19.31 (+114%)	0.345 (+74.2%)	**11.4** (−11.6%)
CALM-PDE	2.46 (+41.4%)	0.201 (+23.3%)	2.25 (+418%)	10.33 (+14.3%)	0.300 (+51.5%)	<u>11.9</u> (−7.75%)
MoLa-PDE (Static)	<u>2.16</u> (+24.1%)	<u>0.178</u> (+9.20%)	1.43 (+229%)	9.92 (+9.73%)	<u>0.225</u> (+13.6%)	15.00 (+16.3%)
MoLa-PDE	**1.74**	**0.163**	**0.434**	9.04	**0.198**	12.90

latent motion is beneficial for both transported localized structures and evolving sharp interfaces. Figure 2 further examines how the benefit of latent motion changes with interface sharpness and transport strength on Sharp Transport (***ST***). For ***ST-Rigid***, moving supports outperform static supports in all nine settings, with relative difference ranging from 28.3% to 104.2%. As transport becomes faster, the improvement generally increases: from slow to fast transport, the differences are $40.9\% \rightarrow 97.7\% \rightarrow 99.3\%$ for smooth interfaces, $36.1\% \rightarrow 95.0\% \rightarrow 104.2\%$ for medium interfaces, and $28.3\% \rightarrow 80.6\% \rightarrow 93.3\%$ for sharp interfaces. The static-support error grows markedly with transport strength, whereas the moving-support error remains comparatively stable. ***ST-Shear*** provides a more challenging nonuniform-transport setting, where different spatial locations undergo different displacements. Unlike rigid transport, the dependence on transport strength is less uniform, reflecting the additional difficulty introduced by spatially varying deformation. Nevertheless, moving supports improve all settings, with substantially larger gains in the sharp, strong-transport regime than for smooth interfaces under the same strong shear. This pattern suggests that adaptive latent motion is particularly useful when sharp structures undergo substantial spatially nonuniform displacement. Fig. 2 also visualizes the learned support motion under rigid and shear transport. In the rigid case, the latent support moves coherently in the dominant transport direction, while under shear the displacement varies across locations following the spatially varying flow. These patterns suggest that the learned latent motion can recover the underlying direction and structure of the transport dynamics.

PDE Benchmark Results. Table 1 compares our method with existing surrogates across six PDE benchmarks. Overall, introducing latent-coordinate motion consistently improves the same static backbone on all systems and gives competitive performance among the compared methods. Notably, the static variant is already competitive on most benchmarks, supporting the effectiveness of the underlying geometry-structured latent architecture. Introducing coordinate motion consistently improves this strong backbone, further highlighting the contribution of the moving latent representation.

The results reveal three main trends. ***First***, the largest gains from latent motion appear on transport-dominated dynamics. On Advection, the error decreases from 1.43 to 0.434, corresponding to nearly a 70% reduction over the static counterpart. Motion also clearly improves Burgers and Navier–Stokes, consistent with the importance of advective transport and evolving coherent structures in these systems, and suggesting that the moving representation can benefit nonlinear transport and deformation beyond simple spatial translation. The improvement on Shallow Water is more moderate, as its dynamics couple transport with wave propagation. ***Second***, motion also improves Diffusion–Reaction from 0.178 to 0.163, despite the absence of an explicit advective term. This suggests that moving supports can provide benefits beyond material transport, potentially by adapting the spatial allocation of latent representation as localized structures evolve. ***Third***, on the irregular cylinder-flow benchmark, motion reduces the error from 15.0 to 12.9, although AROMA and CALM-PDE remain more accurate. This suggests that latent motion remains beneficial even on irregular geometries, but its advantage can be limited when the dynamics are strongly influenced by complex boundary interactions and wake structures, where additional geometry- or boundary-aware modeling may be required. ***Taken together***, these results suggest that the primary advantage of moving latent supports is not a uniform increase in model capacity, but a better alignment between the latent state and PDE dynamics when spatial information must be transported or adaptively redistributed over time.

Ablation Analysis. To understand the contribution of the key design choices in our framework, we ablate two main components. ***First***, we examine the geometry-structured operator by replacing the combined polynomial–Fourier basis with polynomial-only or Fourier-only variants. ***Second***, we study the structured latent motion mechanism by replacing the potential-based and Hodge-inspired motion with an MLP that directly predicts coordinate displacements. All variants otherwise use the same backbone and training protocol, allowing us to isolate the effect of each design choice.

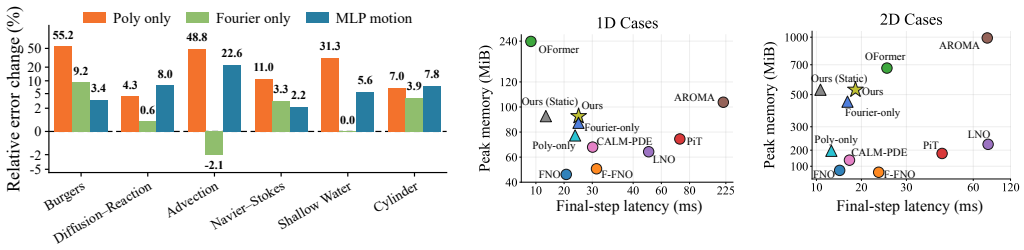

Figure 3: Ablation and computational efficiency analysis. **Left:** Relative error change of the polynomial-only, Fourier-only, and direct MLP-motion variants with respect to the full model across PDE benchmarks; **Right:** Final-step inference latency versus peak memory usage for 1D and 2D settings. Circles denote baselines, the star denotes the full MoLa-PDE, and triangles denote our static and basis-ablation variants.

As shown in Fig. 3 (left), retaining only the low-order polynomial terms leads to substantial degradation on several benchmarks, while the Fourier-only variant remains closer to the full model. However, the two variants are not capacity-matched. The polynomial basis contains only first- and second-order terms, with $P = 3$ in 1D and 6 in 2D, whereas the Fourier basis provides more components across multiple frequencies and spatial directions, with $P = 6$ in 1D and 25 in 2D. Thus, the results should not be interpreted as showing that Fourier components are intrinsically more important. Interestingly, the Fourier-only variant slightly outperforms the full basis on Advection (-2.1%), suggesting that the optimal balance between polynomial and Fourier components can be task-dependent. Designing more PDE-adaptive basis functions that better accommodate the moving latent supports is therefore an important direction for future work. The motion ablation shows a more consistent trend. Replacing the structured motion with direct MLP displacement prediction increases the error on all six benchmarks, with the largest degradation on Advection. This supports explicitly structuring the coordinate dynamics rather than predicting the motion of each latent state independently.

Efficiency and Resource Usage. We evaluate the computational efficiency of the proposed MoLa-PDE in terms of parameter count, peak memory usage, and inference time, and compare it with representative baseline models. Fig. 3 (right) summarizes the main efficiency results, with the full comparison provided in Appendix Table E2. The proposed MoLa-PDE remains competitive in inference speed across both 1D and 2D settings compared with efficient baselines. Its additional computational cost mainly arises from processing evolving spatial geometry and updating latent coordinates, but this enables the model to adapt its representation dynamically and leads to consistent prediction improvements over the static counterpart. Meanwhile, the lighter polynomial- and Fourier-only variants reduce parameter count and memory usage, revealing a clear accuracy–efficiency trade-off with the performance degradation observed in the ablation study. This suggests that improving basis expressiveness under a limited basis size P is an important direction for further efficiency gains. Exploring more compact neural basis functions that retain strong spatial expressiveness with fewer basis components will be an important direction for future work.

5 Conclusion

We introduced MoLa-PDE, a moving-latent neural PDE surrogates, which extend latent-space PDE modeling by jointly evolving spatial coordinates and latent features. The proposed framework combines a geometry-structured continuous operator with a potential-based structured motion mechanism, allowing latent content and spatial support to coevolve throughout autoregressive prediction. Across controlled synthetic problems, moving supports consistently improve prediction accuracy for transported, deforming, and sharp localized structures. On diverse PDE benchmarks, latent motion consistently improves the corresponding static model and outperforms all compared baselines on most dynamic systems. Ablations validate the key design choices, while efficiency results show that the accuracy gains come with moderate overhead and competitive inference efficiency. More broadly, our results suggest that allowing the latent geometry itself to evolve provides a useful degree of freedom for compact PDE representations, particularly when spatial information must be continuously transported or redistributed. We discuss the current limitations of MoLa-PDE in Appendix F. Future work will investigate more expressive basis functions under limited sizes, as well as more efficient mechanisms for modeling moving latent geometry and complex boundary interactions.

AI USE STATEMENT

The authors used generative AI tools during the preparation and experimental execution of this work.

Specifically, OpenAI's ChatGPT (GPT-5.6 Sol) was used for language refinement, including correcting grammatical errors, improving the readability of the manuscript, and retrieving literature to help identify potentially relevant prior work. OpenAI Codex (GPT-5.6) and Anthropic Claude (Claude Opus 5) were used to assist with writing, debugging, and optimizing the experimental code.

All AI-assisted text and code were thoroughly reviewed and verified by the authors. The authors take full responsibility for the scientific content, experimental results, citations, and final content of this work.

REFERENCES

Benedikt Alkin, Andreas Fürst, Simon Lucas Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In *Advances in Neural Information Processing Systems*, 2024.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

Harsh Bhatia, Gregory Norgard, Valerio Pascucci, and Peer-Timo Bremer. The helmholtz-hodge decomposition—a survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(8): 1386–1404, 2012.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

Junfeng Chen and Kailiang Wu. Positional knowledge is all you need: Position-induced transformer (PiT) for operator learning. In *International Conference on Machine Learning*, 2024.

Franco Flandoli, Eliseo Luongo, et al. *Stochastic partial differential equations in fluid mechanics*, volume 2330. Springer, 2023.

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. In *ICML 2022 2nd AI for Science Workshop*, 2022.

Jan Hagnberger, Daniel Musekamp, and Mathias Niepert. Calm-pde: Continuous and adaptive convolutions for latent space modeling of time-dependent pdes. *Advances in Neural Information Processing Systems*, 2025.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: a general neural operator transformer for operator learning. In *International Conference on Machine Learning*, 2023.

Peiyan Hu, Yue Wang, and Zhi-Ming Ma. Better neural PDE solvers through data-free mesh movers. In *International Conference on Learning Representations*, 2024.

Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. Diffusionpde: Generative pde-solving under partial observation. *Advances in Neural Information Processing Systems*, 2024.

Shudong Huang, Wentao Feng, Chenwei Tang, Zhenan He, Caiyang Yu, and Jiancheng Lv. Partial differential equations meet deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 36(8):13649–13669, 2025.

Danial Khatamsaz, Raymond Neuberger, Arunabha M Roy, Sina Hossein Zadeh, Richard Otis, and Raymundo Arróyave. A physics informed bayesian optimization approach for material design: application to niti shape memory alloys. *npj Computational Materials*, 9(1):221, 2023.

Sanghyeon Kim and Eunbyung Park. Smpconv: Self-moving point representations for continuous convolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

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

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.

Leon Lapidus and George F Pinder. *Numerical solution of partial differential equations in science and engineering*. John Wiley & Sons, 1999.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023a. ISSN 2835-8856.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for PDE surrogate modeling. In *Advances in Neural Information Processing Systems*, 2023b.

Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in Neural Information Processing Systems*, 2020.

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, 2023c.

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

Yuxuan Liu, Jingmin Sun, Xinjie He, Griffin Pinney, Zecheng Zhang, and Hayden Schaeffer. PROSE-FD: A multimodal PDE foundation model for learning multiple operators for forecasting fluid dynamics. In *Neurips 2024 Workshop Foundation Models for Science: Progress, Opportunities, and Challenges*, 2024.

Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. In *International Conference on Machine Learning*, 2024.

Suryanarayana Maddu, Dominik Sturm, Bevan L Cheeseman, Christian L Mueller, and Ivo F Sbalzarini. Stencil-net for equation-free forecasting from data. *Scientific reports*, 13(1):12787, 2023.

Eike H Müller and Robert Scheichl. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. *Quarterly Journal of the Royal Meteorological Society*, 140(685):2608–2624, 2014.

Raphaël Pestourie, Youssef Mroueh, Chris Rackauckas, Payel Das, and Steven G Johnson. Physics-enhanced deep surrogates for partial differential equations. *Nature Machine Intelligence*, 5(12): 1458–1465, 2023.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Lukas Prantl, Benjamin Ummenhofer, Vladlen Koltun, and Nils Thuerey. Guaranteed conservation of momentum for learning particle-based fluid dynamics. *Advances in Neural Information Processing Systems*, 2022.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

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

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, 2020.

Louis Serrano, Thomas X Wang, Etienne Le Naour, Jean-Noël Vittaut, and Patrick Gallinari. Aroma: Preserving spatial structure for latent pde modeling with local neural fields. *Advances in Neural Information Processing Systems*, 2024.

Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José M Hernández-Lobato, Richard E Turner, and Emile Mathieu. On conditional diffusion models for pde simulations. *Advances in Neural Information Processing Systems*, 2024.

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*, 2022.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 2020.

Artur Toshev, Jonas A. Erbesdobler, Nikolaus A. Adams, and Johannes Brandstetter. Neural SPH: Improved neural modeling of lagrangian fluid dynamics. In *International Conference on Machine Learning*, 2024.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *International Conference on Learning Representations*, 2023.

Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023. ISSN 0045-7825.

Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Yogesh Verma, Markus Heinonen, and Vikas Garg. Climode: Climate and weather forecasting with physics-informed neural odes. In *International Conference on Learning Representations*, 2024.

Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. *Advances in Neural Information Processing Systems*, 2024.

Hao Wei, Aleksandra Franz, Bjoern List, and Nils Thuerey. Inc: An indirect neural corrector for auto-regressive hybrid pde solvers. *Advances in Neural Information Processing Systems*, 2026.

Mingqiang Wei, Zeyong Wei, Haoran Zhou, Fei Hu, Huajian Si, Zhilei Chen, Zhe Zhu, Jingbo Qiu, Xuefeng Yan, Yanwen Guo, et al. Agconv: Adaptive graph convolution on 3d point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9374–9392, 2023.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning*, 2024.

Tailin Wu, Takashi Maruyama, Qingqing Zhao, Gordon Wetzstein, and Jure Leskovec. Learning controllable adaptive simulation for multi-resolution physics. In *International Conference on Learning Representations*, 2023.

Bo Xu, Xinliang Liu, and Lei Zhang. Dilated convolution neural operator for multiscale partial differential equations. *Journal of Computational and Applied Mathematics*, 461:116408, 2025. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2024.116408.

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

A Related Works

Neural PDE Surrogates. Neural PDE surrogates provide efficient data-driven alternatives to conventional numerical solvers by learning to approximate PDE solution operators from simulation data. Recent studies employ a variety of architectures to model spatial interactions in physical fields. Convolutional and spectral methods exploit local convolutions or structured transforms to efficiently mix information across spatial domains and are commonly applied to dense Eulerian fields on regular grids, with extensions to multiscale dynamics and complex geometries (Tran et al., 2023; Li et al., 2023c; Rahman et al., 2023; Tripura & Chakraborty, 2023; Xu et al., 2025). Kernel- and message-passing methods construct interactions between spatial samples using neighborhood relations, relative coordinates, or learned continuous kernels, and naturally accommodate meshes, point clouds, and other irregular discretizations (Li et al., 2020; Pfaff et al., 2021; Brandstetter et al., 2022; Liu-Schiaffini et al., 2024). Attention-based methods instead learn data-dependent interactions between spatial states, providing flexible long-range coupling across both structured grids and irregularly sampled domains (Li et al., 2023a; Hao et al., 2023; Li et al., 2023b; Wu et al., 2024). These architectural families are not mutually exclusive, and recent models increasingly combine multiple spatial interaction mechanisms.

Latent-Space PDE Modeling. Orthogonal to these architectural choices, another line of work reduces the cost of temporal prediction by evolving PDE dynamics in a compressed latent space. These models follow an encode–process–decode paradigm, compressing the observed field into a lower-dimensional representation, propagating the dynamics in the latent space, and decoding predictions back to the physical domain. Recent approaches construct such latent representations in different forms. Many works use a fixed-size sequence of latent tokens whose spatial organization is represented implicitly or through learned embeddings. LNO (Wang & Wang, 2024) maps geometric samples into a learnable latent sequence through cross-attention, while UPT (Alkin et al., 2024) compresses mesh- or particle-based observations into latent tokens without imposing a grid- or particle-based latent structure. AROMA (Serrano et al., 2024) further constructs geometry-aware latent tokens that preserve local spatial information, while still evolving the dynamics as a compact token sequence. Another line retains an explicit spatial organization in the latent representation. CALM-PDE (Hagnberger et al., 2025) encodes physical fields onto learned spatial query points and performs temporal evolution on this compact spatial support using continuous convolutions. Such spatial supports, however, remain fixed throughout the temporal rollout.

The proposed MoLa-PDE builds on the latter spatial perspective but treats the latent coordinates themselves as part of the recurrent state. Latent features and their spatial support therefore evolve jointly, allowing a fixed number of latent states to adapt their spatial allocation to the changing PDE dynamics.

Moving-Mesh and Lagrangian PDE Modeling. Moving-mesh and Lagrangian approaches provide an alternative to fixed spatial representations by allowing spatial coordinates to evolve together with the underlying dynamics. Moving-mesh methods dynamically redistribute computational nodes to better resolve evolving solution structures, while retaining these nodes as part of the physical discretization (Wu et al., 2023; Hu et al., 2024; Zhang et al., 2024). Lagrangian methods instead represent the system using particles or material points whose coordinates evolve with the physical motion, and have been combined with learned graph-based simulators for fluids and other dynamical systems (Ummenhofer et al., 2020; Sanchez-Gonzalez et al., 2020; Prantl et al., 2022; Toshev et al., 2024).

The proposed MoLa-PDE also evolves spatial coordinates over time, but the moving points have a fundamentally different role. In moving-mesh methods, the moving nodes constitute the computational mesh on which the physical solution is directly represented. In Lagrangian methods, the moving particles or material points have explicit physical identities, whose trajectories follow the underlying material motion. In contrast, the moving points in MoLa-PDE form a compact learned latent support and do not directly represent either the physical discretization or material particles. This distinction provides two advantages. First, temporal evolution is performed on a much smaller set of latent states rather than on the full physical discretization. Second, unlike physical mesh nodes or material particles, the latent states are not tied to persistent physical identities or prescribed material trajectories. Their coordinates can therefore adapt according to the representational demands of the

evolving solution, for example by following moving or deforming structures and reallocating latent capacity toward dynamically important regions. In this sense, MoLa-PDE combines the efficiency of compressed latent evolution with dynamic spatial adaptation, while allowing the support motion to be learned according to evolving solution structures rather than fixed physical identities.

B Additional Method Details

This section provides additional details of the proposed moving-latent framework that are omitted from the main text for clarity. We describe the basis construction, architecture configuration, and training procedure used in our experiments.

B.1 Polynomial–Fourier Basis

We provide the concrete basis specifications used by the geometry-structured continuous operator in our experiments. For a source–target pair, we normalize the boundary-aware relative displacement by the nominal support spacing,

$$\xi = \frac{\Delta_\Omega(Q_i, P_j)}{\ell}, \qquad \ell = N_s^{-1/d},$$

where N_s denotes the number of source states and d is the spatial dimension. The source features predict the coefficients associated with a fixed set of localized polynomial and Fourier basis functions.

We use Gaussian localization factors

$$e_p(\xi) = \exp\left(-\frac{\kappa_p^2\|\xi\|_2^2}{2}\right), \qquad e_\kappa(\xi) = \exp\left(-\frac{\kappa^2\|\xi\|_2^2}{2}\right),$$

with $\kappa_p = 0.5$ and $\mathcal{K} = \{0.5, 1, 2\}$. Accordingly, the basis family used in our experiments is

$$\begin{aligned}\mathcal{B}_d(\xi) = \{e_p(\xi)\} &\cup \{e_p(\xi)\xi_a\}_{a=1}^d \cup \{e_p(\xi)\xi_a\xi_b\}_{1\le a\le b\le d} \\ &\cup \left\{e_\kappa(\xi)\sin(2\pi\kappa\, u^\top\xi),\ e_\kappa(\xi)\left[\cos(2\pi\kappa\, u^\top\xi) - 1\right]\right\}_{\kappa\in\mathcal{K},\ u\in\mathcal{U}_d}.\end{aligned} \tag{8}$$

The polynomial envelope controls the locality of the smooth low-order terms, while each Fourier scale jointly determines its spatial frequency and localization. The sine and centered-cosine terms vanish at zero displacement, and all basis parameters are predefined and fixed during training.

1D basis. In one dimension, there is only one independent spatial direction. Thus, we have a $P = 9$ basis functions: one localized constant, one linear and one quadratic polynomial term, and three localized sine and three centered-cosine terms. The corresponding expansion can be written as

$$\begin{aligned}\mathcal{B}_{1\mathrm{D}}(\xi) = e_p(\xi)\left(c_0 + c_1\xi + c_2\xi^2\right) \\ + \sum_{\kappa\in\{0.5,1,2\}} e_\kappa(\xi)\left[a_\kappa\sin(2\pi\kappa\xi) + b_\kappa\big(\cos(2\pi\kappa\xi) - 1\big)\right].\end{aligned} \tag{9}$$

2D basis. In two dimensions, we use four spatial directions $\mathcal{U} = \{(1,0),\ (0,1),\ \frac{(1,1)}{\sqrt{2}},\ \frac{(1,-1)}{\sqrt{2}}\}$. to capture both axis-aligned and oblique variations while keeping the directional basis compact. With the three frequency scales, this gives $4 \times 3 \times 2 = 24$ directional Fourier modes. The polynomial part consists of one localized constant, two linear terms, and three quadratic terms. Thus, these form a $P = 30$ basis:

$$\begin{aligned}K_{2\mathrm{D}}(\xi) = e_p(\xi)\left(c_0 + c_x\xi_x + c_y\xi_y + c_{xx}\xi_x^2 + c_{xy}\xi_x\xi_y + c_{yy}\xi_y^2\right) \\ + \sum_{\substack{\kappa\in\mathcal{K}\\ u\in\mathcal{U}}} e_\kappa(\xi)\left[a_{\kappa,u}\sin(2\pi\kappa\, u^\top\xi) + b_{\kappa,u}\left(\cos(2\pi\kappa\, u^\top\xi) - 1\right)\right].\end{aligned} \tag{10}$$

The same fixed basis set is used by the geometry-structured operator throughout the encoder, latent evolver, motion global interaction, and decoder, while the learnable coefficient predictors and aggregation parameters are independent across modules.

Table B1: Architecture configurations used in the 1D and 2D experiments.

Configuration	1D	2D
Encoder support sizes	{256, 64, 8}	{1024, 256, 16}
Encoder feature widths	{16, 32, 64}	{32, 64, 128}
Encoder heads	{2, 4, 4}	{2, 4, 8}
Encoder neighborhood sizes	{16, 16, 32}	{16, 16, 32}
Latent states M	8	16
Latent width C	64	128
Evolver heads	8	8
Decoder heads	4	4
Decoder neighborhood size	2	4
Maximum local displacement	0.25	0.25

B.2 Architecture Details

We provide the architecture configurations used for the 1D and 2D experiments. We use the same encoder–evolver–decoder formulation described in Sec. 3.3 throughout, while adjusting the number of latent states, feature widths, and neighborhood sizes by spatial dimension. Table B1 provides the detailed architecture configurations used in the 1D and 2D experiments. The encoder configurations are listed in stage order from the physical domain toward the final latent representation. Support sizes denote the number of target spatial locations at each encoder stage, while neighborhood sizes denote the number of source states used for each target. The maximum local displacement is measured in the normalized domain $[0, 1]^d$.

B.3 Training Details

We train all models end-to-end through autoregressive rollout. The initial encoded latent state is also decoded during training. The objective therefore includes the reconstruction error at t_0 and the prediction errors at subsequent rollout steps. To progressively increase the difficulty of long-horizon prediction, we use a rollout curriculum in which the number of supervised trajectory states grows smoothly over training. Let E denote the total number of epochs and e the current epoch. We define the curriculum factor as $\rho(e) = \frac{1}{2}\left[1 + \tanh\left(\frac{e/E - 0.5}{0.3}\right)\right]$, and increase the rollout length accordingly until reaching the full prediction horizon.

We apply Gaussian jitter to the moved latent support during training to improve robustness to small perturbations of the latent geometry. After the motion update, the coordinates used for subsequent feature evolution are perturbed as

$$\widetilde{Q}_{t+1,i} = \Pi_\Omega\left(Q_{t+1,i} + \sigma_{\text{jitter}}\,\ell\,\epsilon_i\right), \qquad \epsilon_i \sim \mathcal{N}(0, I),$$

where $\ell = M^{-1/d}$ is the nominal latent spacing and $\sigma_{\text{jitter}} \in [0.01, 0.05]$ depending on the dataset. Jitter is enabled only during training and is disabled for validation and inference.

All models are trained for 500 epochs using AdamW with weight decay 10^{-4}. We select the initial learning rate from $[10^{-4}, 10^{-3}]$ depending on the dataset and use a cosine learning-rate schedule updated after each optimizer step.

C Dataset Details

This section provides additional details of the controlled synthetic problems and PDE benchmark datasets used in our experiments, including data generation, spatial and temporal resolutions, and train/test splits.

C.1 Controlled Synthetic Problems

We construct three controlled two-dimensional synthetic problems to study different settings in which moving latent supports may be beneficial: *Smooth Transport* (***T***), *Sharp Interface Propagation* (***S***),

Sharp Transport (***ST***). All problems are defined on the periodic domain $\Omega = [0, 1)^2$ over $t \in [0, 1]$, with snapshots saved every $\Delta t = 0.05$, resulting in 21 frames per trajectory. Each setting contains 500 trajectories, using the first 400 for training and the remaining 100 for testing. The model operates on 64×64 observations in all cases.

Smooth Transport (*T*). ***T*** is designed to represent localized smooth information undergoing spatially nonuniform transport. The initial field is constructed from a random superposition of $K \in \{3, 4, 5\}$ periodic Gaussian packets,

$$u_0(x, y) = \sum_{j=1}^{K} a_j \exp\left(-\frac{d_{x,j}^2 + d_{y,j}^2}{2\sigma_j^2}\right), \tag{11}$$

where the packet centers, amplitudes, and widths are independently randomized. We use $a_j \sim \mathcal{U}(0.55, 1.0)$ and $\sigma_j \sim \mathcal{U}(0.07, 0.12)$, with periodic distances used to avoid discontinuities across the domain boundary. The dynamics follow the linear advection equation

$$\partial_t u + \mathbf{v} \cdot \nabla u = 0, \qquad \mathbf{v}(x, y) = \big(0.25 \sin(2\pi y),\, 0\big). \tag{12}$$

This makes particles at different vertical locations undergo different horizontal displacements, while the dynamics retain an exact characteristic solution and avoid confounding effects from more complex two-dimensional flow topology. Because y remains constant along characteristics, the solution is generated analytically as

$$u(x, y, t) = u_0([x - 0.25t \sin(2\pi y)] \bmod 1,\, y)\,. \tag{13}$$

Sharp Interface Propagation (*S*). ***S*** focuses on the evolution of a sharp interface whose motion is governed by normal propagation rather than material advection. Initial geometries are generated from low-frequency periodic Fourier fields,

$$g(x, y) = \sum_{j=1}^{4} a_j \sin\big(2\pi \mathbf{k}_j^\top (x, y) + \theta_j\big)\,, \tag{14}$$

where four distinct modes are randomly sampled from

$$\mathcal{K} = \{(1, 0), (0, 1), (1, 1), (1, -1), (2, 0), (0, 2), (2, 1), (1, 2)\}.$$

The amplitudes a_j is sampled from $U(0.5, 1.0)$ and then normalized by $\|\mathbf{k}_j\|_2$. Phases θ_j is sampled from $U(0, 2\pi)$. The low-frequency construction provides smooth, irregular, and periodic geometries while avoiding highly oscillatory initial contours. After normalizing the Fourier field, we sample a target foreground fraction $q \sim U(0.20, 0.35)$ and set the threshold as $\tau = \mathrm{quantile}(g, 1 - q)$. Then we produce an irregular periodic binary region by $M(x, y) = \mathbf{1}[g(x, y) > \tau]$. This produces an irregular periodic binary region whose area is controlled by q. For each accepted region, we compute a periodic signed-distance field ϕ_0, with $\phi_0 < 0$ inside the foreground and $\phi_0 > 0$ outside. The observed sharp field is then constructed as

$$u_0(x, y) = \frac{1}{2}\left[1 - \tanh\left(\frac{\phi_0(x, y)}{\epsilon}\right)\right].$$

The Fourier field determines the interface geometry, while interface sharpness is controlled by $\epsilon = 1/128$. This construction therefore yields sharp interfaces with controlled sharpness and diverse shapes and spatial locations. The interface is then evolved according to the level-set equation

$$\partial_t \phi + c(x, y, t) \|\nabla \phi\|_2 = 0, \tag{15}$$

where the spatially and temporally varying normal speed is defined as

$$c(x, y, t) = 0.25\,[1 + 0.40 \sin(4\pi x) + 0.30 \cos(4\pi y)] \cos(\pi t). \tag{16}$$

The spatial modulation induces nonuniform interface deformation, while the temporal factor gradually reverses the propagation direction after the midpoint of the trajectory. This prevents the interface from expanding monotonically across the entire domain and produces a richer sequence of shape changes.

Sharp Transport (*ST*). ***ST*** is divided into two variants, ***ST-Rigid*** and ***ST-Shear***, which share the same class of sharp initial geometries but differ in the underlying transport dynamics. Both use the same sharp interface construction as ***S***. We consider three interface widths, $\epsilon \in \left\{\frac{8}{128}, \frac{2}{128}, \frac{1}{128}\right\}$, corresponding to smooth, medium, and sharp transitions. Changing ϵ modifies only the interface width while keeping the underlying geometry unchanged. For transportation, ***ST-Rigid*** applies a coherent translation to the sharp structure according to $\partial_t u + \mathbf{c} \cdot \nabla u = 0$, where $\mathbf{c} = s\mathbf{d}$. d is a fixed transport direction and $s \in \{0.5, 1.0, 2.0\}$ controls the transport strength. Since the velocity is constant, each frame is generated by exact periodic backtracing,

$$u(x, y, t) = u_0\left([x - sd_x t] \bmod 1,\ [y - sd_y t] \bmod 1\right).$$

Combining three sharpness levels and three transport strengths gives nine controlled settings. ***ST-Shear*** replaces the global translation with a spatially varying shear flow,

$$\partial_t u + \mathbf{v}(y) \cdot \nabla u = 0, \qquad \mathbf{v}(y) = \big(s \sin(2\pi y),\, 0\big). \tag{17}$$

We solve this using the same closed solutions as Eq. (13). We again combine $s \in \{0.25, 0.50, 0.75\}$ and three interface widths to produce nine settings.

C.2 PDE Benchmark Details

We consider five benchmark systems from PDEBench Takamoto et al. (2022) and an additional cylinder-flow dataset Pfaff et al. (2021) on an irregular mesh.

- **1D Burgers.** We use the 1D viscous Burgers dataset governed by $\partial_t u + u\,\partial_x u = \nu\,\partial_{xx} u$, with viscosity $\nu = 10^{-3}$. The system combines nonlinear advection and diffusion and provides a standard benchmark for evolving steep solution structures.
- **1D Advection.** The 1D Advection dataset follows $\partial_t u + \beta\,\partial_x u = 0$, with transport velocity $\beta = 1.0$. It represents a simple transport-dominated system in which the solution is advected through the spatial domain.
- **1D Reaction–Diffusion.** We use the 1D Reaction–Diffusion dataset governed by $\partial_t u = \nu\ \partial_{xx} u + R(u)$, with diffusion and reaction parameters $\nu = 0.5$. This system combines local reaction dynamics with spatial diffusion and provides a non-transport-dominated benchmark.
- **2D Shallow Water.** The 2D Shallow Water dataset follows the shallow-water equations for the evolution of fluid height and horizontal momentum, $\partial_t \mathbf{q} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{S}(\mathbf{q})$, with the physical parameters following the PDEBench configuration. The dataset contains coupled transport and wave-propagation dynamics.
- **2D Navier–Stokes.** We use the 2D Navier–Stokes dataset with viscosity $\nu = 10^{-5}$. In vorticity form, the dynamics can be written as $\partial_t \omega + \mathbf{v} \cdot \nabla \omega = \nu \Delta \omega + f$, where $\mathbf{v}$ is the incompressible velocity field. This benchmark contains nonlinear advection, diffusion, and evolving vortical structures.
- **Cylinder Flow.** We additionally consider flow around a cylinder, defined on an irregular computational mesh. The underlying flow dynamics are also governed by the incompressible Navier–Stokes equation. The diameter and position of the cylinder change with each sample. This dataset evaluates the model under irregular spatial discretization, obstacle geometry, and wake dynamics.

Dataset statistics, including spatial resolution, trajectory length, and dataset-specific model settings, are summarized in Table C1.

D Baseline and Implementation Details

We compare against a broad set of physical- and latent-space neural PDE surrogates. We follow the implementations and training protocols adopted in CALM-PDE (Hagnberger et al., 2025). For 1D Burgers, 2D Navier–Stokes, and Cylinder Flow, we directly report the baseline results from CALM-PDE under the corresponding benchmark settings. For the remaining datasets, we reproduce the baselines following the same implementations and training protocols. We briefly summarize the baseline models below:

Table C1: Dataset statistics and model input configurations. Reduction denotes the spatial and temporal subsampling factors. T, $\mathbf{X}$, and C denote time steps, spatial resolution, and the number of channels, respectively.

Dataset	Raw data ($T \times \mathbf{X}$)	# Train/Test	Reduction ($\mathbf{X}/T$)	Processed data ($T \times \mathbf{X} \times C$)	Rollout
1D Burgers	201×1024	$9000/1000$	$1/5$	$41 \times 1024 \times 1$	40
1D Advection	201×1024	$9000/1000$	$1/5$	$41 \times 1024 \times 1$	40
1D Reaction–Diffusion	101×1024	$9000/1000$	$1/5$	$21 \times 1024 \times 1$	20
2D Shallow Water	$101 \times 128 \times 128$	$900/100$	$2/5$	$21 \times 64 \times 64 \times 1$	20
2D Navier–Stokes	$20 \times 64 \times 64$	$1000/200$	$1/1$	$20 \times 64 \times 64 \times 1$	19
Cylinder Flow	$30 \times (1732–2059)$	$1000/100$	$1/2$	$15 \times (1732–2059) \times 3$	14

- **Fourier Neural Operator (FNO).** (Li et al., 2023c) FNO models PDE dynamics directly on the physical grid using Fourier-domain integral operators. We use the standard spectral-convolution architecture and train the model autoregressively with a curriculum that gradually increases the prediction horizon during training.
- **Factorized Fourier Neural Operator (F-FNO).** (Tran et al., 2023) F-FNO replaces the multidimensional Fourier transform in FNO with factorized spectral operations along individual spatial dimensions, reducing the computational cost of spectral mixing. We use the same autoregressive curriculum strategy as for FNO.
- **Operator Transformer (OFormer).** (Li et al., 2023a) OFormer uses attention-based encoder and decoder modules to decouple input and output spatial locations, allowing predictions on arbitrary query coordinates. Temporal evolution is performed in the learned latent representation. Following the original implementation and CALM-PDE evaluation protocol, we train OFormer with a progressively increasing rollout horizon.
- **Position-Induced Transformer (PIT).** (Chen & Wu, 2024) PIT constructs its attention weights from spatial distances and evolves the solution through a fixed latent grid of reduced spatial resolution. The latent grid preserves the dimensionality of the physical domain while using fewer spatial points.
- **Latent Neural Operator (LNO).** (Wang & Wang, 2024) LNO compresses the input field into a fixed-size latent representation using position-based attention. A Transformer evolves the latent representation in time, and an attention-based decoder maps it back to the physical domain.
- **AROMA.** (Serrano et al., 2024) AROMA compresses the physical field into a fixed latent representation through Perceiver-style cross-attention. Latent temporal dynamics are modeled with a Transformer and denoising diffusion, and a neural-field decoder reconstructs the solution at physical query locations.
- **CALM-PDE.** (Hagnberger et al., 2025) CALM-PDE is a latent-space PDE surrogate based on continuous convolutions. Its encoder progressively maps the physical field onto a compact set of latent query points, the latent representation is evolved recurrently, and a continuous decoder reconstructs the solution at requested spatial locations.

E Additional Results

This section provides the complete numerical results for the controlled synthetic problems, additional computational efficiency measurements, and visualizations of the learned latent-support motion.

E.1 Numerical Results of Controlled Synthetic Problems

Table E1 reports the complete numerical results for the controlled synthetic problems discussed in the main text, including ***T***, ***S***, ***ST-Rigid***, and ***ST-Shear***. For ***ST***, we report all combinations of interface sharpness and transport strength.

Table E1: Performance on the controlled synthetic problems. Errors are relative ℓ_2 errors. Percentages denote the relative error difference with respect to Motion. For ***ST***, smaller ϵ corresponds to sharper interfaces, while larger s corresponds to faster transport.

Problem	Dynamics	Sharpness	s	Static ↓	Motion ↓	Rel. diff. ↑
T	Shear transport	Smooth	0.25	0.0232	**0.0209**	11.0%
S	Interface propagation	1/128	0.25	0.1210	**0.0986**	22.7%
ST-Rigid	Rigid transport	8/128	0.5	0.0186	**0.0132**	40.9%
		8/128	1.0	0.0259	**0.0131**	97.7%
		8/128	2.0	0.0289	**0.0145**	99.3%
		2/128	0.5	0.0358	**0.0263**	36.1%
		2/128	1.0	0.0509	**0.0261**	95.0%
		2/128	2.0	0.0582	**0.0285**	104.2%
		1/128	0.5	0.0462	**0.0360**	28.3%
		1/128	1.0	0.0652	**0.0361**	80.6%
		1/128	2.0	0.0750	**0.0388**	93.3%
ST-Shear	Shear transport	8/128	0.5	0.0288	**0.0248**	16.1%
		8/128	1.0	0.0322	**0.0289**	11.4%
		8/128	2.0	0.0353	**0.0334**	5.7%
		2/128	0.5	0.0478	**0.0437**	9.4%
		2/128	1.0	0.0606	**0.0523**	15.9%
		2/128	2.0	0.0668	**0.0553**	20.8%
		1/128	0.5	0.0629	**0.0533**	18.0%
		1/128	1.0	0.0778	**0.0640**	21.6%
		1/128	2.0	0.0785	**0.0657**	19.5%

E.2 Complete Computational Efficiency Results

Table E2 reports the complete efficiency measurements underlying Fig. 3 (right). We consider three inference settings. *One-step* measures the cost of advancing the model by a single prediction step. *Final-step* performs the complete autoregressive rollout from the initial condition to the target horizon but returns only the final prediction. For latent-space models that support *"latent skipping"*, intermediate states can remain in the latent space without being decoded. *Full rollout* additionally reconstructs the physical solution at every prediction step and therefore measures the cost of producing the complete trajectory. For each setting, we report inference latency and peak memory usage together with the model parameter count.

E.3 Latent Support Motion Visualization

We provide additional visualizations of the learned latent-support motion on PDE trajectories in Fig. E1 – E6, illustrating how the latent coordinates evolve throughout the rollout. The visualizations reveal three characteristic behaviors of the learned latent motion.

First, the motion patterns depend strongly on the underlying PDE dynamics. Advection and Burgers show coherent support displacements that follow transported structures, while Shallow Water exhibits redistribution around propagating wavefronts. This indicates that MoLa-PDE can learn dynamics-dependent support motion that follows different types of evolving spatial structures.

Second, the gains from latent motion are also spatially dynamics-dependent. They are more localized around fronts or evolving structures in Advection, Burgers, and Shallow Water, whereas Diffusion–Reaction shows more broadly distributed improvements across the domain.

Third, Cylinder Flow exhibits a different pattern: most support displacement occurs early in the rollout, after which the latent geometry remains nearly stationary. This suggests that coordinate evolution mainly reconfigures the initial uniform support, while later dynamics are represented primarily through feature evolution. Overall, the learned motion is representation-driven rather than following a single fixed motion pattern.

Table E2: Inference efficiency and peak memory usage on the 1D and 2D settings with batch size $B = 8$. Latency is the median batch inference time in milliseconds, and peak memory is the allocated memory in MiB. Lower is better for both latency and memory.

			Full Rollout		Final-step		One-step	
Model	#Params	Latent skip	Latency (ms)	Peak (MiB)	Latency (ms)	Peak (MiB)	Latency (ms)	Peak (MiB)
1D Cases								
FNO	549,569	×	20.941	48.6	20.774	46.2	0.578	46.2
F-FNO	665,281	×	31.814	53.9	31.499	50.7	0.836	50.7
OFormer	660,719	✓	9.598	240.7	8.231	238.7	2.671	238.7
PiT	78,861	×	73.241	77.6	73.114	74.5	1.929	74.4
LNO	461,121	×	51.028	67.2	51.007	64.3	1.408	64.3
AROMA	1,748,105	✓	233.939	106.2	189.222	103.8	6.345	103.8
CALM-PDE	568,331	✓	31.485	572.0	30.127	68.0	6.041	68.1
MoLa-PDE	541,629	✓	30.060	116.9	25.146	92.7	1.413	92.7
MoLa-PDE (Static)	541,629	✓	17.482	116.9	13.534	92.7	1.114	92.7
MoLa-PDE (Poly-only)	315,453	✓	27.846	91.2	23.915	77.3	1.238	77.3
MoLa-PDE (Fourier-only)	466,237	✓	29.773	108.3	25.115	87.4	1.374	87.4
2D Cases								
FNO	1,812,161	×	14.882	78.9	15.160	74.3	0.857	74.2
F-FNO	151,361	×	24.098	66.7	23.832	62.7	1.356	62.6
OFormer	1,850,497	✓	31.282	666.0	25.670	663.8	10.272	663.8
PiT	1,249,805	×	37.934	183.9	37.927	179.6	2.390	179.4
LNO	2,847,105	×	93.436	240.8	92.985	236.4	4.922	236.3
AROMA	1,845,505	✓	129.133	996.2	91.116	991.7	7.562	991.7
CALM-PDE	2,230,883	✓	18.836	934.5	17.342	137.9	6.509	137.9
Ours (Motion)	5,202,771	✓	27.240	530.4	18.676	530.4	3.635	530.4
Ours (Static)	5,202,771	✓	19.071	530.4	10.894	530.4	3.199	530.4
Ours (Poly-only)	1,698,003	✓	17.944	196.3	13.317	196.3	1.759	189.5
Ours (Fourier-only)	4,472,611	✓	23.592	456.2	16.902	456.2	3.206	456.2

F LIMITATION

Our moving-latent framework allows latent supports to evolve with the PDE dynamics, enabling the representation to better track changing spatial structures and improve prediction accuracy. Despite these benefits, several limitations remain.

First, the polynomial–Fourier basis relies on a predefined set of basis terms, and increasing its spatial expressiveness generally requires a larger basis size P. As observed in our ablation and efficiency analysis, richer basis families improve prediction accuracy but also increase parameter count and computational cost. Developing more expressive learned or neural basis functions under a smaller P is therefore an important direction for improving the accuracy–efficiency trade-off.

Second, although the latent support evolves dynamically during prediction, its initial configuration is obtained from a fixed uniform encoder support and is not adapted to each input sample. Consequently, the recurrent motion module must perform all subsequent spatial reallocation from the same initial layout. A sample-adaptive initialization could instead allocate latent states according to the structures already present in the initial condition, potentially reducing the burden on later coordinate evolution and improving the representation of highly localized or nonuniform spatial features from the beginning of the rollout.

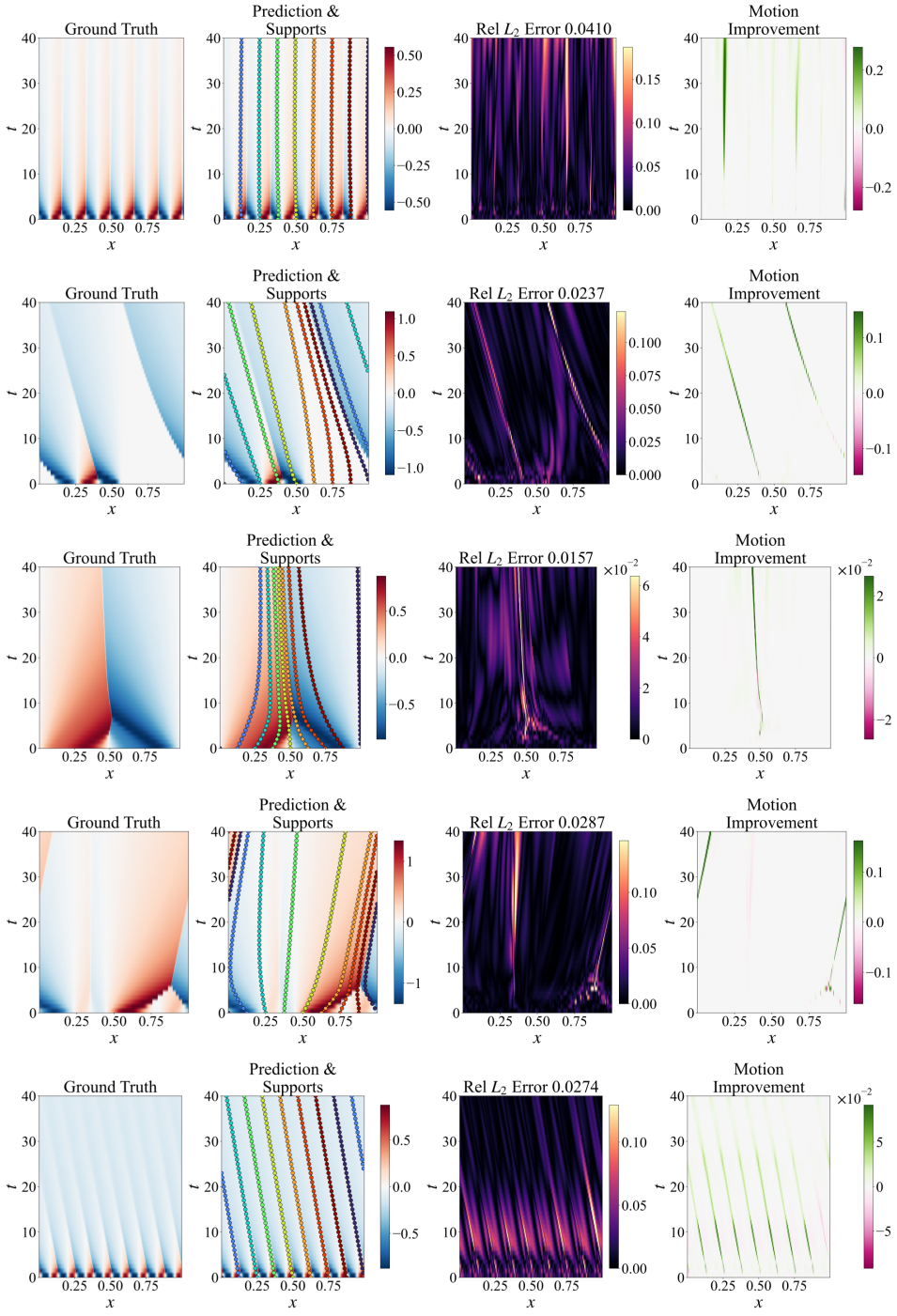

Figure E1: Visualization of learned latent-support motion on *Burgers* dataset. Each row shows a different test example. From left to right, we show the ground-truth trajectory, the MoLa-PDE prediction overlaid with the evolving latent supports, the prediction error, and the improvement induced by latent motion relative to the corresponding static-support model.

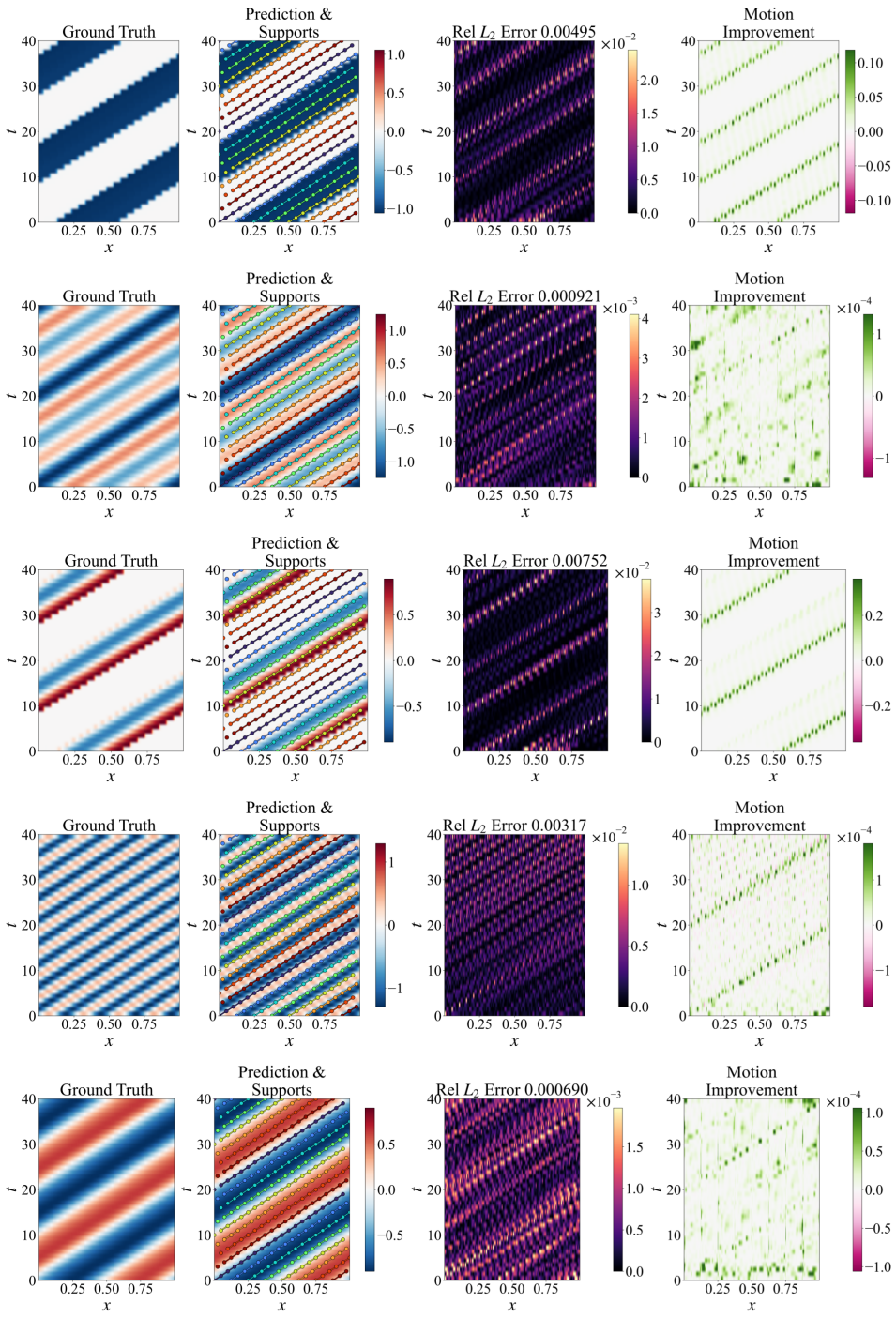

Figure E2: Visualization of learned latent-support motion on *Advection* trajectories. Each row shows a different test example. From left to right, we show the ground-truth trajectory, the MoLa-PDE prediction overlaid with the evolving latent supports, the prediction error, and the improvement induced by latent motion relative to the corresponding static-support model.

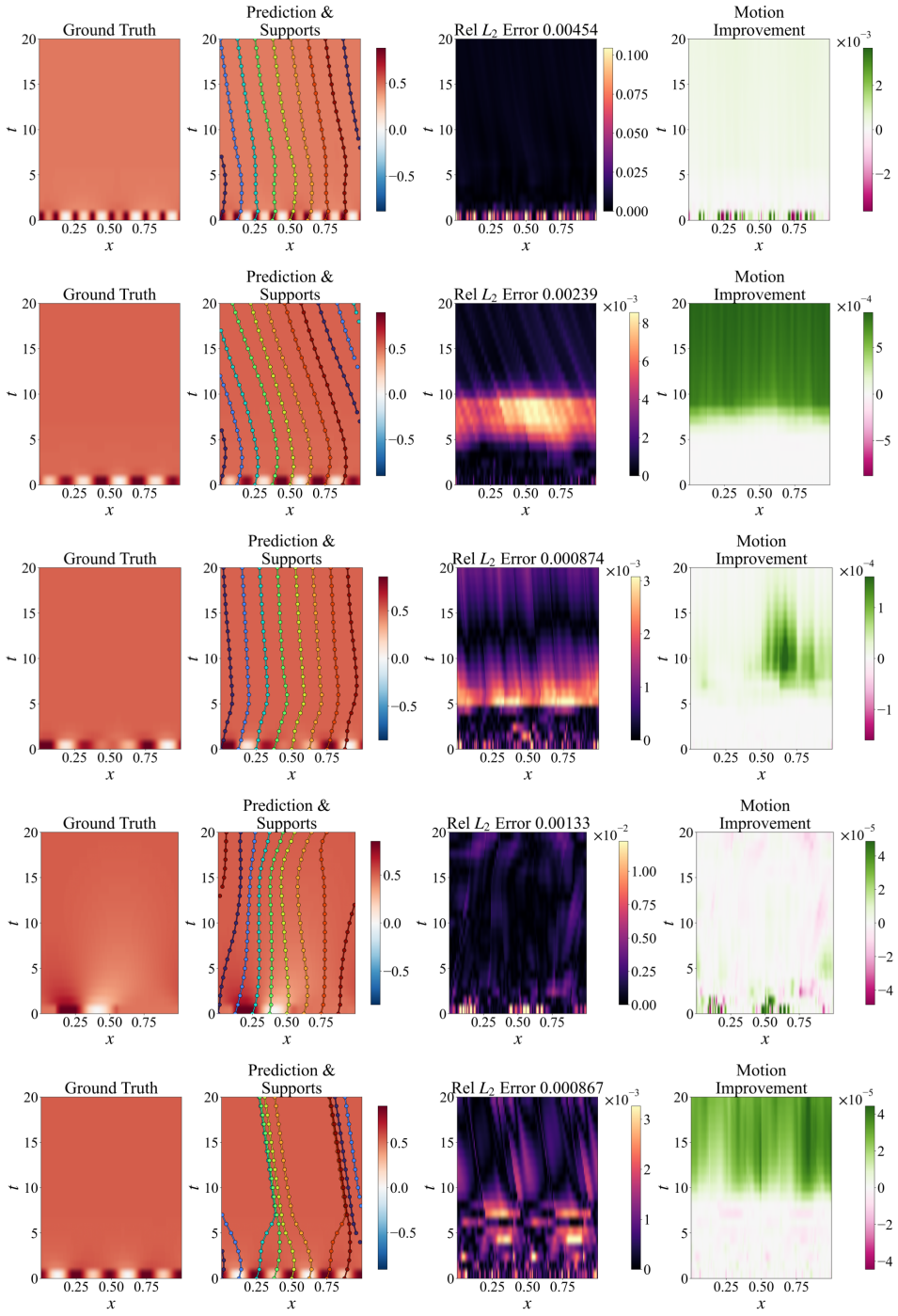

Figure E3: Visualization of learned latent-support motion on *Reaction-Diffusion* dataset. Each row shows a different test example. From left to right, we show the ground-truth trajectory, the MoLa-PDE prediction overlaid with the evolving latent supports, the prediction error, and the improvement induced by latent motion relative to the corresponding static-support model.

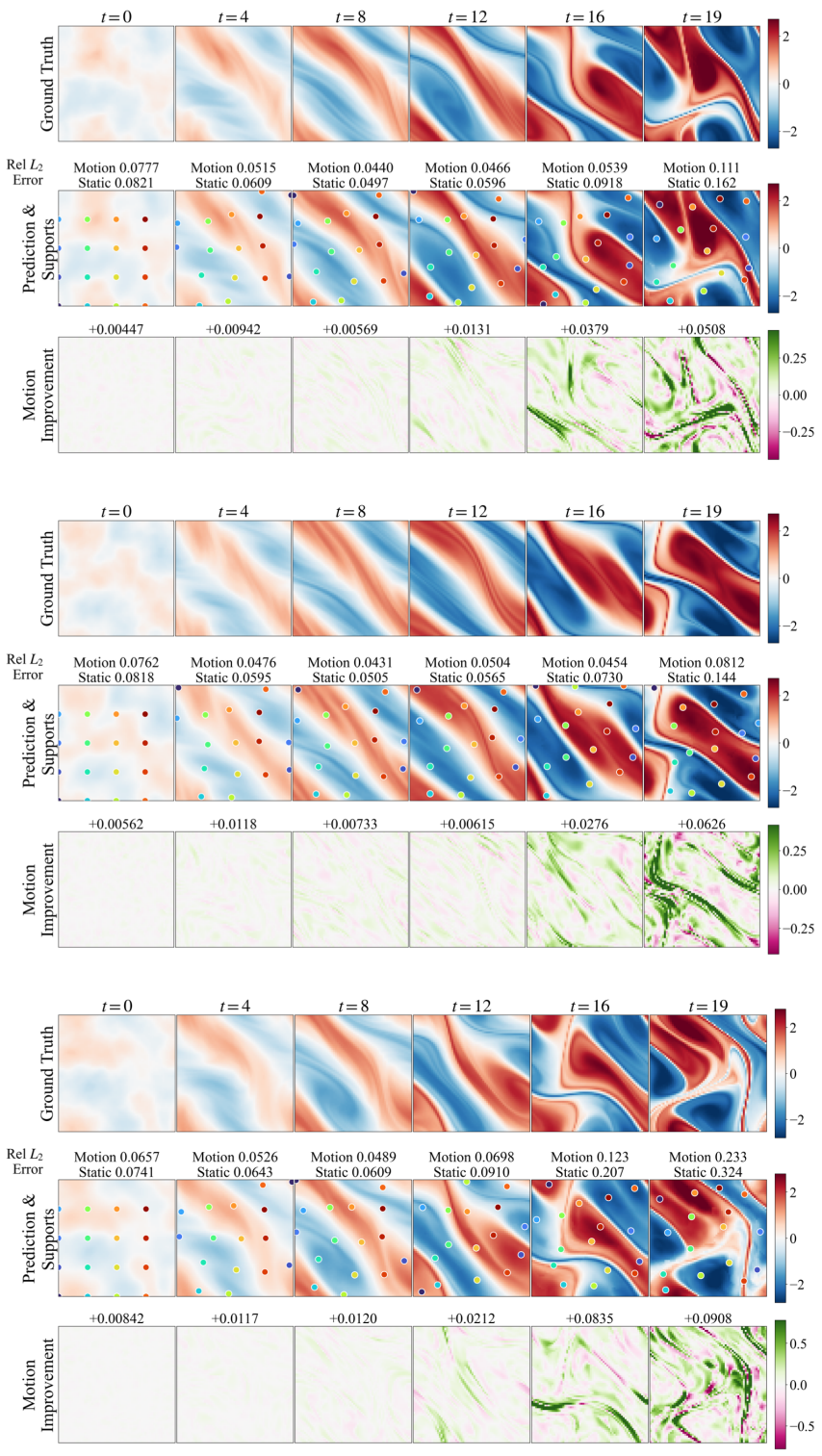

Figure E4: Visualization of learned latent-support motion on *Navier–Stokes* dataset. For each example, columns show selected time steps throughout the rollout. The top row shows the ground-truth solution, the middle row shows the MoLa-PDE prediction overlaid with the evolving latent supports together with the relative L_2 errors of the moving and static variants, and the bottom row visualizes the spatial improvement induced by latent motion.

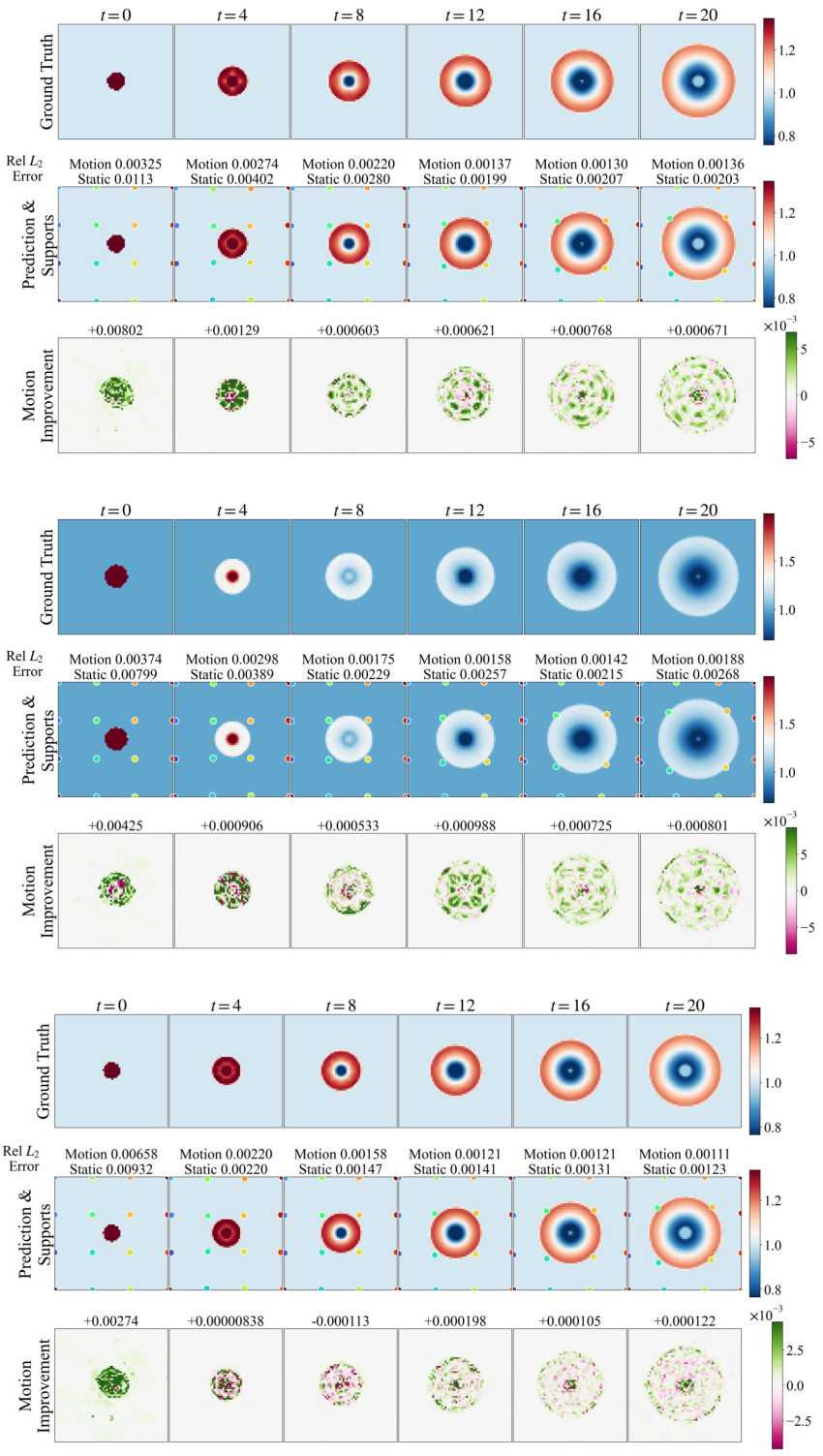

Figure E5: Visualization of learned latent-support motion on *Shallow Water* dataset. For each example, columns show selected time steps throughout the rollout. The top row shows the ground-truth solution, the middle row shows the MoLa-PDE prediction overlaid with the evolving latent supports together with the relative L_2 errors of the moving and static variants, and the bottom row visualizes the spatial improvement induced by latent motion.

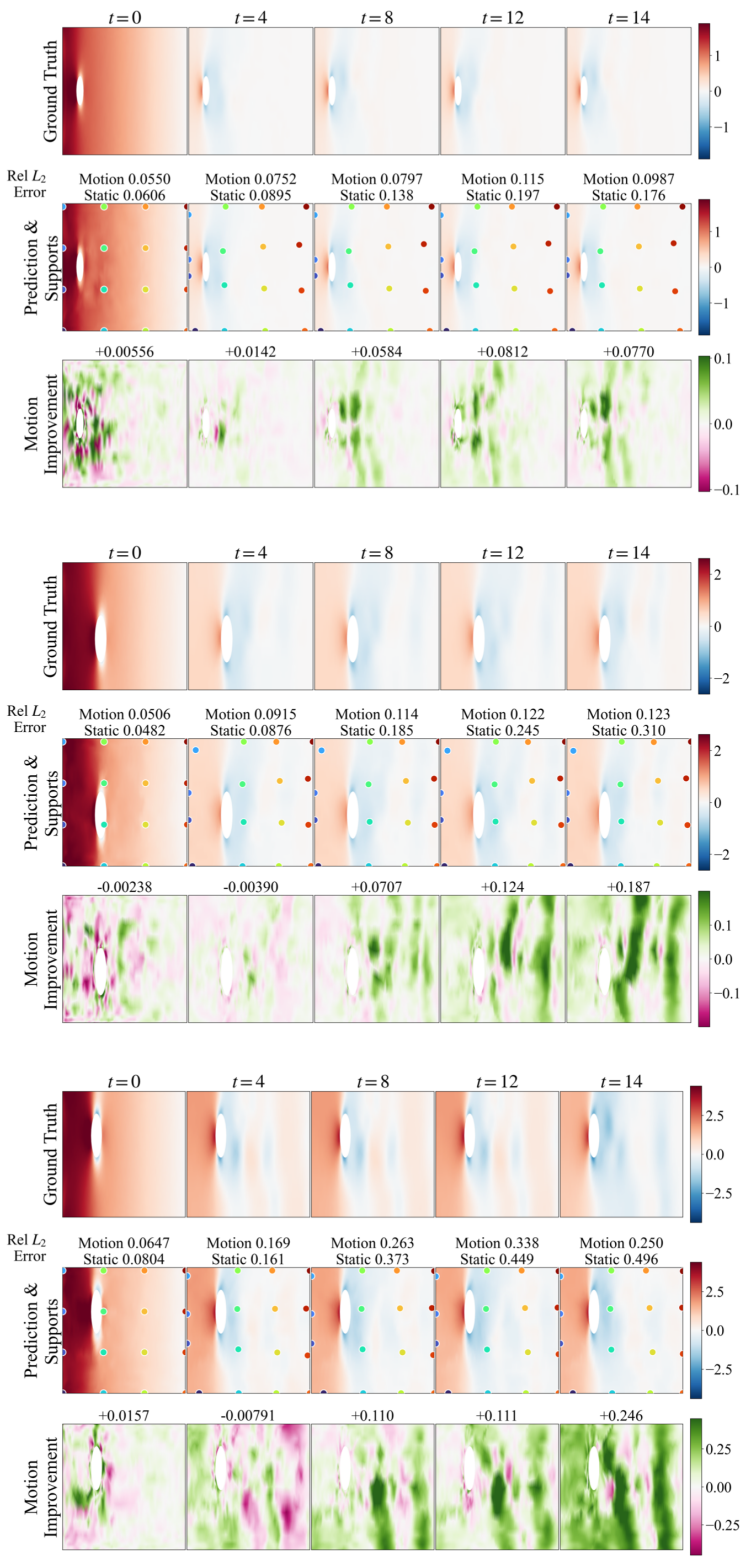

Figure E6: Visualization of learned latent-support motion on *Cylinder Flows* dataset. For each example, columns show selected time steps throughout the rollout. The top row shows the ground-truth solution, the middle row shows the MoLa-PDE prediction overlaid with the evolving latent supports together with the relative L_2 errors of the moving and static variants, and the bottom row visualizes the spatial improvement induced by latent motion.