HOTPOT: OPERATOR TRANSFORMER WITH HETEROGENEOUS PHYSICS EXPERTS FOR LARGE-SCALE PDE PRE-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale PDE pre-training trains a single model on the data from multiple partial differential equations (PDEs). However, adding heterogeneous PDEs with different governing equations, physical variables, and variable interactions does not always improve predictive performance across tasks. Existing PDE mixture-of-experts models mitigate cross-task interference through learned routing, but typically use homogeneous expert architectures, making it difficult to directly reflect differences in state evolution across equations. We introduce HotPOT, an operator transformer with heterogeneous physics experts for large-scale PDE pre-training. HotPOT uses a shared encoder to capture information common across PDEs, while each PDE family uses a neural–numerical expert that pairs a learnable predictor with a fixed numerical update containing no trainable parameters. Each predictor outputs the quantities required by its paired numerical update, which then computes the next state. This design allows different PDEs to share a common encoder while retaining their own learnable predictors and numerical updates. Experiments across five PDE families and six settings show that HotPOT outperforms MoE-POT in all 24 matched-scale comparisons before and after fine-tuning, reducing rollout error by 50.7% on average. Notably, the 17M model also surpasses substantially larger MoE-POT models, and the performance gains extend to zero-shot prediction under unseen physical conditions. Code is available at https://anonymous.4open.science/r/HotPOT-code-1727/.

1 INTRODUCTION

Partial differential equations (PDEs) are fundamental to modeling a wide range of physical systems. The high cost of repeated numerical simulations has motivated learning-based PDE solvers, particularly neural operators (Li et al., 2021; Hmida et al., 2026) that learn solution mappings directly from data. Recent methods have shown that a single model can be trained jointly on data from multiple PDEs, rather than trained separately for each task, and have achieved strong performance across multiple PDE tasks.

However, adding data from heterogeneous PDEs with different governing equations, physical variables, and variable interactions does not always improve performance on existing tasks. For example, the MoE-POT model proposed in Wang et al. (2025) reports increased error on some original tasks when expanding the training mixture of a dense pre-trained model from six to twelve datasets. To mitigate cross-task interference, MoE-POT uses learned routing to select specialized experts. These experts nevertheless use homogeneous architectures, which may be difficult to directly learn inherent physical distinctions in state evolution across equations.

On the other hand, Neural–numerical methods offer a basis for constructing heterogeneous experts by combining learned predictions with explicit numerical operations (Karlbauer et al., 2022; Gopakumar et al., 2026). Different PDEs can require different update operations and prediction quantities. An expert may predict a rate for time integration, an increment for a constrained update, or state components for direct assignment. Pairing a predictor with the corresponding update allows the expert's computation to reflect the requirements of its PDE. This motivates a neural–numerical pre-training paradigm in which each PDE pairs learnable predictions with explicit numerical op-

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.

erations according to its update requirements, while shared history representations integrate the resulting experts into a unified pre-training model.

Specifically, we introduce HotPOT, an operator transformer that combines a shared encoder with heterogeneous neural–numerical experts for large-scale PDE pre-training. Figure 1(a) summarizes the workflow from heterogeneous PDE pre-training to prediction and transfer. The shared encoder processes scalar-field histories across PDEs, while each expert handles system-specific variable coupling and state evolution. Each expert pairs a learnable predictor with a PDE-specific fixed numerical update, which contains no trainable parameters. The predictor outputs the quantities required by its paired numerical update, which then computes the next state. The shared encoder and expert predictors are jointly trained using next-state supervision, bringing the different neural–numerical experts together in a single pre-training model. The Navier–Stokes (NS) example in Fig. 1(b) illustrates the resulting prediction: at the tenth forecast step, HotPOT follows the reference vorticity contours more closely than MoE-POT in the displayed central region.

The main contributions are:

- We propose HotPOT, an operator transformer that leverages PDE-specific experts containing learnable predictors with fixed numerical updates, allowing different equations to use different prediction quantities and update operations.
- We integrate these heterogeneous experts through a shared history encoder into one multi-PDE model. The encoder and predictors are trained end-to-end using only next-state supervision, while variable coupling and state evolution remain system-specific.
- We evaluate HotPOT across five PDE families and six settings. Relative to MoE-POT, HotPOT reduces rollout error by 50.7% on average across 24 matched-scale comparisons before and after fine-tuning, and further improves zero-shot prediction under unseen physical conditions.

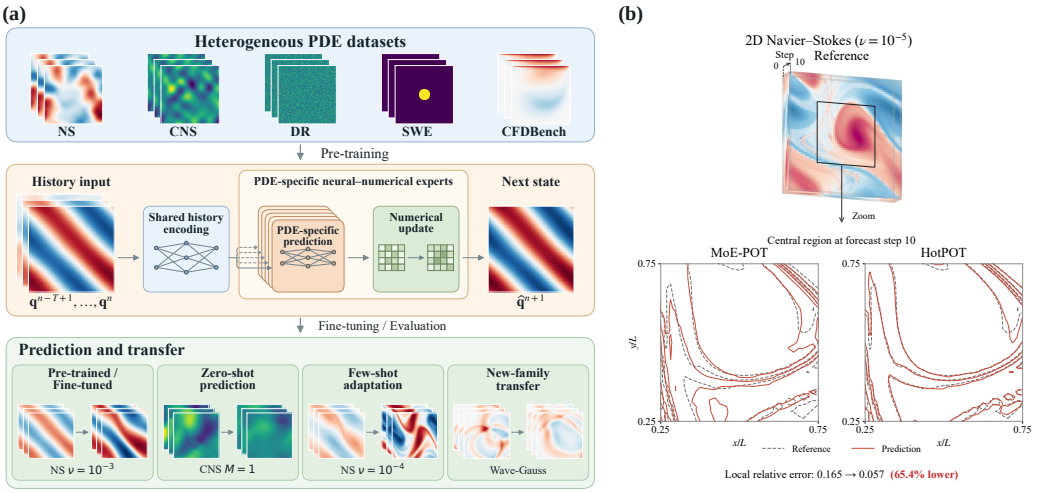

Figure 1: **HotPOT: framework overview and an NS prediction example.** (a) Shared history encoding connects PDE-specific predictor–update pairs for pre-training and transfer. The state is $\mathbf{q}^n$ at step n, with history length $T = 10$. (b) Pre-trained 17M HotPOT and MoE-POT on the same NS test trajectory ($\nu = 10^{-5}$). The boxed region is compared at forecast step 10; gray dashed and red solid curves denote reference and predicted contours, respectively. The relative L_2 error in this region decreases from 0.165 for MoE-POT to 0.057 for HotPOT (65.4% lower). Here ν is viscosity, M is Mach number, and L is domain length.

2 Related Work

Multi-PDE pre-training. Previous studies including MPP (McCabe et al., 2024), DPOT (Hao et al., 2024), Poseidon (Herde et al., 2024), and Walrus (McCabe et al., 2026) learn reusable representations through joint training across PDE systems. Some related approaches propose to better

organize representations around physical variables or scalar fields, as in CoDA-NO (Rahman et al., 2024) and Tadpole (Liu et al., 2026), or condition prediction on equation and system information, as in PDEformer (Ye et al., 2024), PROSE-PDE (Sun et al., 2025), and Unisolver (Zhou et al., 2025). These studies provide the foundation for learning across heterogeneous PDE data. We focus on incorporating PDE-specific prediction and update operations into this shared pre-training framework.

Mixture-of-experts for PDEs. MoE-POT (Wang et al., 2025) combines shared experts with input-dependent routed experts, while NESTOR (Sun et al., 2026a) routes computation at both image and token levels. These methods allocate specialized neural capacity to different inputs through learned routing, while retaining a common expert structure. HotPOT extends expert specialization to the quantities predicted and the operations used to evolve the state. Each PDE family uses a specific neural predictor paired with its corresponding fixed numerical update. A shared history encoder connects these heterogeneous experts within one jointly trained model.

Neural–numerical methods. Neural–numerical methods place learned components within structured updates. Universal differential equations (Rackauckas et al., 2021) and APHYNITY (Yin et al., 2021) augment known dynamics, while learned discretizations and solver-in-the-loop methods learn numerical operators or corrections (Bar-Sinai et al., 2019; Kochkov et al., 2021; Um et al., 2020). FINN (Karlbauer et al., 2022) embeds trainable components in PDE-specific finite-volume structures, and OpsSplit (Gopakumar et al., 2026) combines learned nonlinear operators with fixed finite-difference operators and time integration. CompNO (Hmida et al., 2026) combines pretrained operators with boundary enforcement, and ComPhy (Trenta et al., 2026) couples equation-specific modules. Origo (Sun et al., 2026b) decomposes evolution into global spectral and local constitutive operators, while Serrano et al. (2026) compose learned operators at test time. DISCO (Morel et al., 2025) generates a trajectory-conditioned operator for time integration. Building on this line of work, HotPOT organizes PDE-specific predictor–update pairs as heterogeneous experts for multi-PDE pre-training. Their prediction quantities and update operations differ across PDEs, while the shared encoder and expert predictors are jointly trained using next-state supervision.

3 Problem Setting

We consider joint pre-training over a collection of time-dependent PDE systems. Let $\mathcal{R}$ denote the set of prediction settings, and let $r \in \mathcal{R}$ index one setting. A setting specifies a physical system together with its benchmark configuration, such as the observed variables, equation parameters, geometry, or boundary conditions. Different settings may therefore have different state variables, variable couplings, and state-transition mechanisms.

Each setting r has a trajectory dataset $\mathcal{D}_r$ with C_r observed physical fields. Flattening the spatial grid in a fixed order gives $\mathbf{q}^n_{r,c} \in \mathbb{R}^{N_r}$ for field c at saved step n, where N_r is the number of grid cells and $q^n_{r,c,i}$ is its scalar value at cell i. The operator concat concatenates its entries in the specified order. The full state is the vector $\mathbf{q}^n_r = \mathrm{concat}(\{\mathbf{q}^n_{r,c}\}_{c=1}^{C_r}) \in \mathbb{R}^{N_r C_r}$. Geometry and support channels describe the spatial domain and valid grid locations. They are provided as auxiliary inputs rather than physical fields to be predicted, so they are excluded from C_r and the state vector $\mathbf{q}^n_r$. The pre-training corpus is $\mathcal{D} = \bigcup_{r\in\mathcal{R}} \mathcal{D}_r$.

Let $g_{\mathbf{w}}$ denote the next-state predictor with trainable parameter vector $\mathbf{w}$. The operator stack stacks its indexed row vectors in order. Let $\widehat{\mathbf{q}}^{n+1}_r$ denote the predicted next state. The most recent $T = 10$ saved states form the history matrix $\mathbf{H}^n_r$:

$$\mathbf{H}^n_r = \mathrm{stack}\big(\{(\mathbf{q}^{n-T+j}_r)^\top\}_{j=1}^{T}\big), \qquad \widehat{\mathbf{q}}^{n+1}_r = g_{\mathbf{w}}(\mathbf{H}^n_r; r). \tag{1}$$

Thus $\mathbf{H}^n_r \in \mathbb{R}^{T\times N_r C_r}$ contains states in chronological order. The matrix $\mathbf{H}^n_{r,c} \in \mathbb{R}^{T\times N_r}$ is its field-c block. The argument r identifies the PDE setting. The predictor also uses the geometry and support information, which we omit from the arguments of $g_{\mathbf{w}}$ to keep the notation compact. Each prediction is appended to the history for autoregressive rollout. We use bold lowercase symbols for vectors, bold uppercase symbols for matrices, and calligraphic uppercase symbols for sets or spaces. Indexed braces gather entries in index order. Parentheses group arguments or pairs.

4 METHOD

HotPOT builds its shared encoder and neural predictors using patch encoding, temporal aggregation, and grouped Fourier/local blocks from DPOT and MoE-POT (Hao et al., 2024; Wang et al., 2025). As shown in Fig. 2, HotPOT realizes $g_{\mathbf{w}}$ by connecting PDE-family predictor–update pairs through a shared history encoder.

Let $\tau : \mathcal{R} \to \mathcal{T}$ map settings to the set of expert families. The family $\tau(r)$ is assigned to setting r. Settings from the same PDE family use the corresponding predictor and numerical update program. In particular, the two NS viscosity settings share one NS expert family. The shared encoder has parameter vector $\boldsymbol{\theta}$. For family $e \in \mathcal{T}$, $\boldsymbol{\phi}_e$ parameterizes field coupling and evolution, and $\boldsymbol{\psi}_e$ parameterizes decoding and readout. Their aggregation is $\mathbf{w} = \mathrm{concat}(\boldsymbol{\theta}, \{(\boldsymbol{\phi}_e, \boldsymbol{\psi}_e)\}_{e\in\mathcal{T}})$, where parameter pairs are flattened and concatenated in a fixed family order.

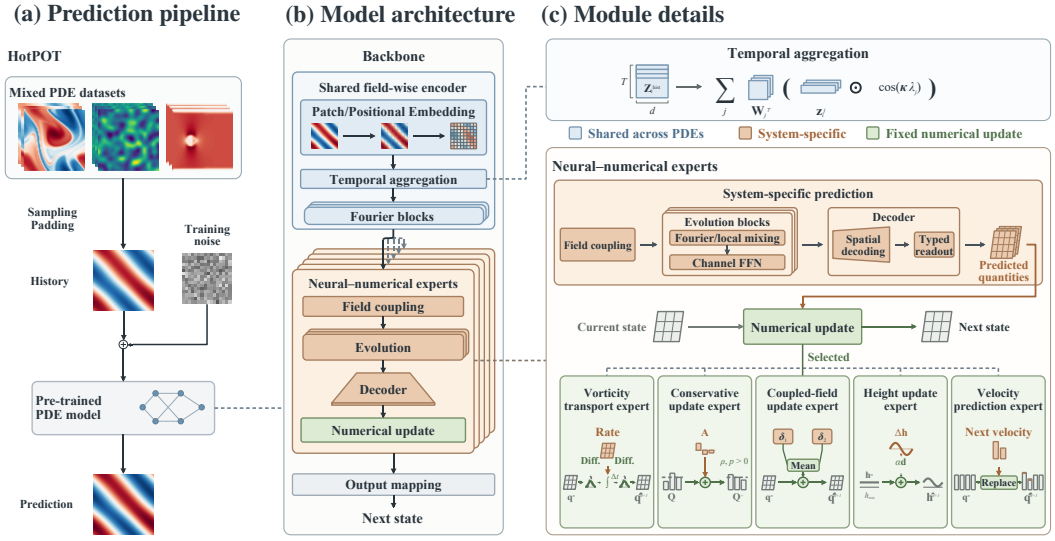

Figure 2: **Architecture of HotPOT. (a)** Prediction pipeline with sampling, padding, and training-time noise. **(b)** A shared encoder connects PDE-specific neural–numerical experts in one model. **(c)** Temporal aggregation and five alternative numerical updates. Blue, orange, green, and gray indicate shared computation, system-specific prediction, fixed numerical updates, and physical states, respectively. Stage-state queries, history statistics, and repeated readouts are omitted. FFN denotes feed-forward network. Field images are illustrative. Notation and module details are given in Appendices A.1 and A.2.

4.1 SHARED FIELD-WISE HISTORY REPRESENTATION

The deterministic function $f_{\mathrm{feat},r,c}$ normalizes field c's history and constructs six input features: normalized field value, two coordinates, boundary distance, geometry support, and field validity. This mapping does not mix physical fields. Each scalar history then passes through the same encoder $f_{\mathrm{enc},\boldsymbol{\theta}}$, with no learned cross-field mixing. At saved time n, let $\mathbf{Z}_{r,c} \in \mathbb{R}^{P\times d_{\mathrm{enc}}}$ denote its representation, where P is the number of spatial patches and d_{enc} the shared width. Then

$$\mathbf{Z}_{r,c} = f_{\mathrm{enc},\boldsymbol{\theta}}\big(f_{\mathrm{feat},r,c}(\mathbf{H}^n_{r,c})\big), \qquad c = 1, \ldots, C_r. \tag{2}$$

A shared patch encoder and positional embedding produce tokens for each history position. Temporal aggregation then combines these tokens within each field. Shared Fourier/local and channel feed-forward network (FFN) blocks produce this feature matrix. The saved-time index n is suppressed. Next-state supervision jointly trains these features across datasets. Appendix A.1 gives the aggregation formula, dimensions, and block equations.

4.2 System-Specific Prediction

Each PDE family's predictor supplies the quantities required by its numerical update: transport rates for NS, conservative-variable actions for CNS, height increments for SWE, concentration-increment proposals for DR, or next-state velocities for CFDB. We denote this prediction space by $\mathcal{U}_r$.

Field coupling and evolution. Let $f_{\mathrm{mix},\boldsymbol{\phi}_{\tau(r)}}$ denote field coupling followed by system-specific evolution. For field c, let $\mu_{r,c}$ and $\sigma_{r,c}$ be the history mean and floored standard deviation over space and the T input frames, and collect them as $\mathbf{s}_{r,c} = (\mu_{r,c}, \log \sigma_{r,c})^\top \in \mathbb{R}^2$. The coupled context $\mathbf{Z}_r^{\mathrm{mix}} \in \mathbb{R}^{P \times d_{\mathrm{priv}}}$, with private width d_{priv}, is

$$\mathbf{Z}_r^{\mathrm{mix}} = f_{\mathrm{mix},\boldsymbol{\phi}_{\tau(r)}}\left(\{(\mathbf{Z}_{r,c}, \mathbf{s}_{r,c})\}_{c=1}^{C_r}\right). \tag{3}$$

Coupling concatenates field representations, applies a 1×1 projection, and adds history-statistic and positional embeddings. System-specific Fourier/local/FFN blocks then produce the context.

Spatial decoding and prediction. The decoder $f_{\mathrm{dec},\boldsymbol{\psi}_{\tau(r)}}$ expands $\mathbf{Z}_r^{\mathrm{mix}}$ to full resolution using a transposed convolution, Gaussian error linear unit (GELU) activation, and pointwise projection. Its readout combines this cached context with a query state $\boldsymbol{\xi}_r^{n,k} \in \mathbb{R}^{d_{\xi,r}}$, representing the current or an intermediate state. Here $k = 1, \ldots, K_r$ indexes queries within the saved interval Δt_r, the time between consecutive saved states. Let $\widehat{\mathbf{u}}_r^{n,k} \in \mathcal{U}_r \subseteq \mathbb{R}^{d_{u,r}}$ denote the neural prediction at query k; $d_{\xi,r}$ and $d_{u,r}$ count the spatial and channel entries of the flattened query and prediction, respectively. Then

$$\widehat{\mathbf{u}}_r^{n,k} = f_{\mathrm{dec},\boldsymbol{\psi}_{\tau(r)}}(\mathbf{Z}_r^{\mathrm{mix}}, \boldsymbol{\xi}_r^{n,k}). \tag{4}$$

The same symbol $\widehat{\mathbf{u}}_r^{n,k}$ represents the different physical quantities listed above. The following section specifies their interpretation, reshaping, and conversion into the next state.

4.3 System-Specific Numerical Execution

Let $f_{\mathrm{upd},\tau(r)}^{\Delta t_r}$ be the PDE family's fixed numerical update over Δt_r. It combines the current state $\mathbf{q}_r^n$ with the neural predictions from Eq. (4) to produce the next-state estimate $\widehat{\mathbf{q}}_r^{n+1}$:

$$\widehat{\mathbf{q}}_r^{n+1} = f_{\mathrm{upd},\tau(r)}^{\Delta t_r}\left(\mathbf{q}_r^n, \{\widehat{\mathbf{u}}_r^{n,k}\}_{k=1}^{K_r}\right). \tag{5}$$

The program has no trainable parameters. For multi-stage updates, it constructs successive query states from earlier predictions and reuses the history context and readout weights. Fixed input/output mappings connect its grid to the observed state layout; the PDE's physical coefficients are implicit in its arguments. Below we give the core updates, omitting setting index r. Let N be the program's grid-cell count, $\mathbf{1}_N$ the all-ones vector, and $\langle\cdot\rangle$ spatial mean. A superscript $+$ denotes a program's updated state. Operator details and update properties are in Appendices A.2–A.3.

Incompressible Navier–Stokes (NS). Here the neural output $\widehat{\mathbf{u}}^{n,k}$ is an effective vorticity transport rate. Let $\boldsymbol{\omega} \in \mathbb{R}^N$ be the current substep vorticity, $\mathbf{f} \in \mathbb{R}^N$ the forcing, ν the viscosity, and δt the substep duration. Write $f_{\mathrm{diff},\nu}^{s}$ for numerical diffusion over duration s. Let $\boldsymbol{\omega}_a$ be the diffused state and $\boldsymbol{\omega}_m$ the midpoint state; $\widehat{\mathbf{r}}_1$ and $\widehat{\mathbf{r}}_2$ are the corresponding rate outputs of Eq. (4) at these two states. The updated vorticity $\boldsymbol{\omega}^+$ is computed in this order:

$$\begin{aligned} \boldsymbol{\omega}_a &= f_{\mathrm{diff},\nu}^{\delta t/2}(\boldsymbol{\omega}), & \boldsymbol{\omega}_m &= \boldsymbol{\omega}_a + \tfrac{\delta t}{2}(\widehat{\mathbf{r}}_1 + \mathbf{f}), \\ \boldsymbol{\omega}^+ &= f_{\mathrm{diff},\nu}^{\delta t/2}(\boldsymbol{\omega}_a + \delta t(\widehat{\mathbf{r}}_2 + \mathbf{f})). \end{aligned} \tag{6}$$

Composing these substeps yields the next saved vorticity state. Transport rates are learned; diffusion and midpoint integration are fixed operations.

Compressible Navier–Stokes (CNS). Reshape the neural output $\widehat{\mathbf{u}}^{n,1}$ as a normalized action matrix $\mathbf{A} \in \mathbb{R}^{N \times 4}$ for density, two momentum components, and total energy. Let $\mathbf{Q} \in \mathbb{R}^{N \times 4}$ be the current conservative state, $\bar{\mathbf{a}} \in \mathbb{R}^4$ the action's channel means, and $f_{\mathrm{scale}}(\mathbf{Q}) \in \mathbb{R}^4$ its conservative-field channel scales; diag forms a diagonal matrix. Denote the increment by $\Delta\mathbf{Q}$, the candidate by

$\mathbf{Q}^\star$, and the updated state by $\mathbf{Q}^+$. A global coefficient $\alpha_{\mathrm{CNS}} \in [0,1]$ enforces positive density and pressure:

$$\begin{aligned} \Delta\mathbf{Q} &= (\mathbf{A} - \mathbf{1}_N \bar{\mathbf{a}}^\top)\,\mathrm{diag}(f_{\mathrm{scale}}(\mathbf{Q})), \qquad \mathbf{Q}^\star = \mathbf{Q} + \Delta\mathbf{Q}, \\ \mathbf{Q}^+ &= \mathbf{Q} + \alpha_{\mathrm{CNS}}(\mathbf{Q}^\star - \mathbf{Q}). \end{aligned} \tag{7}$$

The program restores candidate conservative means before the segment update and converts $\mathbf{Q}^+$ back to primitive variables for the next state. For admissible inputs, global mass, momentum, and energy are preserved in exact arithmetic (Proposition 2).

Shallow-water equation (SWE). The neural output $\widehat{\mathbf{u}}^{n,1} = \Delta\mathbf{h} \in \mathbb{R}^N$ is a height increment. Let $\mathbf{h}^n$ and $\widehat{\mathbf{h}}^{n+1}$ be the current and predicted height vectors, $\mathbf{d}$ the centered increment, and $\alpha \in [0,1]$ a global coefficient enforcing a positive height floor. The update is

$$\mathbf{d} = \Delta\mathbf{h} - \langle \Delta\mathbf{h} \rangle \mathbf{1}_N, \qquad \widehat{\mathbf{h}}^{n+1} = \mathbf{h}^n + \alpha\mathbf{d}. \tag{8}$$

Centering and uniform scaling preserve mass on the uniform grid in exact arithmetic while respecting the height floor for admissible inputs (Proposition 1).

Diffusion–reaction (DR). The neural outputs $\boldsymbol{\delta}_1 = \widehat{\mathbf{u}}^{n,1}$ and $\boldsymbol{\delta}_2 = \widehat{\mathbf{u}}^{n,2}$ are concentration-increment proposals in $\mathbb{R}^{2N}$ for the two fields. The first is queried at the current state $\mathbf{q}^n$; the second at the intermediate state $\boldsymbol{\xi}^{n,2}$. The program constructs this state and then adds the averaged proposals to obtain $\widehat{\mathbf{q}}^{n+1}$:

$$\boldsymbol{\xi}^{n,2} = \mathbf{q}^n + \tfrac{1}{2}\boldsymbol{\delta}_1, \qquad \widehat{\mathbf{q}}^{n+1} = \mathbf{q}^n + \tfrac{1}{2}(\boldsymbol{\delta}_1 + \boldsymbol{\delta}_2). \tag{9}$$

CFDBench (CFDB). For the velocity-prediction tasks in CFDB, reshape the neural output $\widehat{\mathbf{u}}^{n,1}$ as $\widehat{\mathbf{V}} \in \mathbb{R}^{N\times 2}$, containing the two predicted velocity fields. The next velocity state $\mathbf{V}^{n+1}$ is assigned directly:

$$\mathbf{V}^{n+1} = \widehat{\mathbf{V}}. \tag{10}$$

The program retains auxiliary support channels; output mapping restores the observed layout.

4.4 Joint Training and Autoregressive Prediction

Let $\boldsymbol{\pi}$ be the nonnegative setting-weight vector, with $\sum_{r\in\mathcal{R}} \pi_r = 1$, and let $\mathcal{O}_r \subseteq \{1,\dots,C_r\}$ be the set of scored physical-channel indices. We sample settings from the categorical distribution $\mathrm{cat}(\boldsymbol{\pi})$. Sampling from $\mathcal{D}_r$ below means drawing a history–target window from its trajectories. With denominator stabilizer $\epsilon = 10^{-8}$, the expected next-state loss $\ell_{\mathrm{train}}(\mathbf{w})$ supervises the complete predictor–program pair:

$$\ell_{\mathrm{train}}(\mathbf{w}) = \mathbb{E}_{\substack{r\sim\mathrm{cat}(\boldsymbol{\pi}) \\ (\mathbf{H}_r^n, \mathbf{q}_r^{n+1})\sim\mathcal{D}_r}} \frac{1}{|\mathcal{O}_r|} \sum_{c\in\mathcal{O}_r} \frac{\|\widehat{\mathbf{q}}_{r,c}^{n+1} - \mathbf{q}_{r,c}^{n+1}\|_2}{\|\mathbf{q}_{r,c}^{n+1}\|_2 + \epsilon}. \tag{11}$$

The norms run over valid spatial cells of each field vector, excluding padded cells. Scored channels are weighted equally within each setting. Gradients through executed numerical arithmetic train the selected expert and shared encoder without intermediate labels. During rollout, each prediction enters the history window and replaces its oldest frame.

New-family adaptation retains this objective, reusing the shared encoder with a newly initialized system-specific predictor and a configured fixed numerical update program, as detailed in Appendix A.7.

5 Experiments

We follow MoE-POT's benchmark (Wang et al., 2025) for joint pre-training and task-specific fine-tuning, complemented by ablations, within-family generalization, and transfer to a new PDE family.

Datasets and evaluation. Six settings cover five PDE families: NS at viscosities $\nu = 10^{-5}$ and $\nu = 10^{-3}$ (Li et al., 2021), PDEBench CNS, SWE, and DR (Takamoto et al., 2022), and CFDBench velocity prediction (Luo et al., 2023). We retain the benchmark's ten-frame inputs, padding, and masking, and evaluate full-horizon autoregressive predictions using mean relative L_2 error (L2RE). Joint pre-training and task-specific fine-tuning run for 1000 and 200 epochs, respectively, with final checkpoints used for evaluation.

Baselines and model scales. We compare eight baseline methods across fifteen configurations: six separately trained models FNO, UNet, FFNO, GK-T, Oformer, and GNOT and three scales each of pre-trained FNO, DPOT, and MoE-POT. We also report fine-tuned DPOT and MoE-POT results. Baseline results and activated-parameter counts are taken from MoE-POT Table 1. HotPOT-T/S use approximately 17M/90M activated parameters, matching MoE-POT-T/S. Appendices A.4 and A.5 provide detailed evaluation, training, and resource settings.

5.1 Main Experiments

Table 1: Full-horizon mean L2RE on six PDE settings (↓). CNS lists Mach number M and shear viscosity η, with bulk viscosity $\zeta = \eta$. Bold: best within each training block; blue: best overall. HotPOT uses final E1000/E200 checkpoints. Baselines follow MoE-POT Table 1 (Wang et al., 2025), with matched splits, histories, horizons, resolutions, and metrics. Dashes denote errors above 20.

Model	Activated Params	NS benchmarks		PDEBench			CFDBench
		NS (10^{-5})	NS (10^{-3})	CNS$(0.1, 0.01)$	SWE	DR	
Small models							
FNO	0.5M	**0.156**	0.0128	0.170	0.00440	0.120	0.00761
UNet	25M	0.198	0.0245	0.357	0.0521	0.0971	0.0209
FFNO	1.2M	0.161	0.0256	0.183	0.00458	0.161	0.0990
GK-T	1.1M	0.260	0.0148	0.919	0.0453	**0.0120**	0.419
Oformer	1.8M	0.289	**0.00319**	**0.161**	0.00474	0.991	**0.00444**
GNOT	2.2M	0.590	0.316	0.533	**0.00199**	0.930	0.0216
Pre-trained							
FNO-T	10M	0.191	0.0245	0.0859	11.0	0.530	0.00601
FNO-S	30M	0.157	0.0225	0.357	0.184	0.385	0.00460
FNO-M	150M	0.141	0.00730	–	0.0104	**0.0112**	–
DPOT-T	7.5M	0.107	0.0155	0.0168	0.00631	0.0577	0.00673
DPOT-S	30.8M	0.0688	0.00781	0.0244	0.00392	0.0367	0.00870
DPOT-M	122M	0.0569	0.00708	0.0224	0.00247	0.0288	0.0113
MoE-POT-T	17M	0.0682	0.00768	0.0105	0.00640	0.0411	0.00529
MoE-POT-S	90M	0.0552	0.00583	0.00959	0.00289	0.0342	0.00448
MoE-POT-M	288M	0.0528	0.00570	0.00914	0.00299	0.0300	0.00513
HotPOT-T	17M	0.0315	0.00325	0.00671	0.00177	0.0150	**0.00219**
HotPOT-S	90M	**0.0299**	**0.00323**	**0.00623**	**0.00166**	0.0326	0.00225
Fine-tuned							
DPOT-T	7.5M	0.0700	0.00725	0.0168	0.00313	0.0289	0.00391
DPOT-S	30.8M	0.0502	0.00635	0.0238	0.00315	0.0215	0.00586
DPOT-M	122M	0.0424	0.00593	0.0221	0.00260	0.0175	0.00653
MoE-POT-T	17M	0.0456	0.00493	0.00746	0.00305	0.0188	0.00330
MoE-POT-S	90M	0.0361	0.00376	0.00742	0.00251	0.0182	0.00313
MoE-POT-M	288M	0.0351	0.00388	0.00744	0.00193	0.0140	0.00398
HotPOT-T	17M	0.0215	**0.000481**	0.00583	0.00162	**0.00163**	0.00208
HotPOT-S	90M	**0.0193**	0.000483	**0.00536**	**0.00154**	0.00436	**0.00202**

Pre-training and fine-tuning. Across six settings, two activated-parameter scales, and both training stages, HotPOT outperforms MoE-POT in all 24 matched-scale comparisons (Table 1). Mean relative error reductions are 47.1% after pre-training and 54.3% after fine-tuning, or 50.7% across both stages. For example, after fine-tuning, HotPOT-T reduces L2RE relative to MoE-POT-T by 90.3% on NS at $\nu = 10^{-3}$ and 91.3% on DR. Appendices A.9–A.10 provide paired field visualizations and space–time error plots.

Accuracy with fewer activated parameters. The 17M HotPOT-T model also outperforms the 90M and 288M MoE-POT models on every setting in both stages. Across the twelve setting–stage comparisons, its mean relative error reductions are 47.7% and 46.5%, respectively, while using approximately 81.1% and 94.1% fewer activated parameters.

Parameter accounting. Each HotPOT-T active path uses 16.77–17.18M parameters, comprising the 2.012M shared encoder and the selected system-specific predictor. Detailed parameter accounting and numerical query counts are provided in Appendix A.5.

5.2 Ablation Studies

Full denotes HotPOT-T. All models use 17M configurations and fixed E1000 checkpoints; Mean equally weights six full-horizon L2REs.

Ablations of cross-PDE sharing. We test whether to share history encoding and where sharing belongs relative to private coupling. *Learned private history* gives each PDE family a separate trainable encoder, retaining expert predictors and numerical updates, to test history sharing. *Late-shared evolution* moves shared processing after private field coupling to test this sharing boundary.

In Table 2, Full lowers mean error by 3.2% relative to private history while reducing stored parameters by 7.6%. Private history performs better on SWE and slightly better on DR. Late sharing raises mean error from 0.0101 to 0.0286, worsening five settings. The placement controls in Appendix A.6 favor history processing before private coupling over late evolution processing; sharing at the history stage further improves mean accuracy and reduces storage.

Table 2: Cross-PDE sharing ablations. Full-horizon L2RE; bold marks minima before rounding.

Model	NS 10^{-5}	NS 10^{-3}	CNS	SWE	DR	CFDB	Mean
Full HotPOT	**0.0315**	**0.00325**	**0.00671**	0.00177	0.0150	0.00219	**0.0101**
Learned private history	0.0322	0.00420	0.00790	**0.000962**	**0.0149**	0.00222	0.0104
Late-shared evolution	0.0331	0.00413	0.00941	0.00421	0.119	**0.00217**	0.0286

Ablations of numerical experts. We test the predictor–update interface across PDE families and, specifically for NS, whether its numerical update provides a benefit beyond repeated neural refinement. *Direct-state predictor* (Direct) replaces predicted quantities and numerical execution with residual state prediction, retaining the neural architecture, inputs, masks, and nominal readout-call counts. This changes the interface and stage-state queries together. *Generic iteration* targets the NS expert, replacing its numerical update with repeated, state-dependent residual predictions using shared readout weights. Other expert paths retain their updates.

In Table 3, Full lowers mean error from 0.0324 to 0.0101 relative to direct prediction, improving five settings; Direct is slightly better on CFDB. Relative to Generic iteration, Full lowers NS error by 1.5% at $\nu = 10^{-5}$ and 19.7% at $\nu = 10^{-3}$. These comparisons support the paired predictor–update interface for aggregate accuracy and NS numerical execution over generic learned iteration on both viscosity settings. Appendix A.6 details the controls and temporal analyses; Appendix A.8 reports physical diagnostics.

Table 3: Numerical-expert ablations. Full-horizon L2RE; bold marks minima before rounding.

Model	NS 10^{-5}	NS 10^{-3}	CNS	SWE	DR	CFDB	Mean
Full HotPOT	**0.0315**	**0.00325**	**0.00671**	**0.00177**	**0.0150**	0.00219	**0.0101**
Direct-state predictor	0.0525	0.00625	0.0104	0.00190	0.122	**0.00211**	0.0324
Generic iteration	0.0320	0.00404	0.00720	0.00177	0.0150	0.00234	0.0104

5.3 Within-Family Generalization

We evaluate pre-trained 17M models at the original resolutions under unseen conditions within supported PDE families.

Zero-shot prediction. For CNS, M is the Mach number and η the shear viscosity. The bulk viscosity is $\zeta = \eta$ in the reported datasets (Takamoto et al., 2022). Fixed pre-training checkpoints predict CNS at $(M, \eta) = (1, 10^{-2})$ and CFDB boundary-condition, geometry, and physical-property subsets within the cavity and cylinder families, without target training or checkpoint selection. We report L2RE over ten forecast steps, denoted H10, with case-uniform masked aggregation for CFDB. This differs from the full-horizon main benchmark. Table 4 shows that HotPOT lowers error relative to MoE-POT in all seven groups, including a 55.5% CNS reduction.

Table 4: Zero-shot H10 L2RE (↓) using fixed 17M pre-training checkpoints without target adaptation. CFDB uses case-uniform masked aggregation.

	CNS	Cavity			Cylinder		
Model	$(M, \eta) = (1, 10^{-2})$	BC	Geo†	Prop	BC	Geo†	Prop
HotPOT-T	**0.189**	**0.00568**	**0.0169**	**0.00467**	**0.00119**	**0.00180**	**0.000577**
MoE-POT-T	0.423	0.0276	0.0439	0.0116	0.00802	0.0117	0.00223

BC, Geo, and Prop denote boundary-condition, geometry, and physical-property subsets. †Each geometry subset contains one held-out case within its respective cavity or cylinder family.

Few-shot adaptation. Pre-trained models adapt to NS ($\nu = 10^{-4}$) and CNS ($(M, \eta) = (1, 10^{-2})$) using 16, 64, or 256 training trajectories. On 200 test trajectories per target, HotPOT achieves lower full-rollout L2RE in all six comparisons, with error decreasing across budgets (Table 5). Rollout validation selects checkpoints; Appendix A.4.1 details the protocol.

Table 5: Within-family adaptation: model-feedback L2RE on 200 test trajectories per target (NS/CNS: 20/11 steps). Full-rollout validation selects checkpoints during 500-epoch adaptation.

	NS $\nu = 10^{-4}$			CNS $(M, \eta) = (1, 10^{-2})$		
Model / shots	16	64	256	16	64	256
HotPOT-T	**0.124**	**0.0964**	**0.0819**	**0.0983**	**0.0599**	**0.0365**
MoE-POT-T	0.304	0.206	0.108	0.365	0.153	0.0735

5.4 Transfer to a New PDE Family

Wave-Layer and Wave-Gauss (Herde et al., 2024) belong to one wave-equation family absent from both pre-training mixtures. HotPOT jointly trains its transferred encoder and a new scalar predictor. Scratch uses the same architecture, initial predictor weights, residual updates, and training protocol, but a random encoder. This comparison tests whether shared encoder pre-training benefits adaptation to a new PDE family.

At 16, 32, 64, and 128 shots, rollout validation selects checkpoints evaluated on 200 held-out trajectories per dataset using mean per-step L2RE. Encoder transfer outperforms both baselines in all eight comparisons. Error reductions are 11.9–39.8% relative to MoE-POT full fine-tuning; relative to Scratch, they are 6.5–8.4% on Wave-Layer and 0.6–16.4% on Wave-Gauss, supporting transfer of the pre-trained history representation. Appendix A.7 gives results (Table 11), protocol, and splits.

6 Discussion and Conclusion

HotPOT combines shared field-history encoding with PDE-specific neural–numerical experts, allowing different equations to use different prediction quantities and fixed update operations. Across six pre-training settings on regular grids, HotPOT outperforms MoE-POT at matched activated-parameter scales before and after fine-tuning. Ablations support the sharing and prediction–execution designs, while transfer to an unseen wave-equation family supports encoder reuse with limited target data. System-specific experts increase storage as the model covers more PDEs, and adapting to a new family requires training a predictor and configuring its numerical update. Multi-stage updates can also increase inference cost through repeated neural readouts. Future work will explore reusable experts, automatic numerical-program composition, and efficient execution for broader PDE collections.

AI USE STATEMENT

Generative AI tools assisted with language editing.

REPRODUCIBILITY STATEMENT

The appendix specifies the data interface, model dimensions, system-specific numerical updates, optimization schedule, parameter accounting, and checkpoint selection. Code, configurations, and fixed evaluation scripts are available in the anonymous repository linked in the abstract.

REFERENCES

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. doi: 10.1073/pnas.1814058116.

John Crank and Phyllis Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67, 1947. doi: 10.1017/S0305004100023197.

Vignesh Gopakumar, Ander Gray, Dan Giles, Lorenzo Zanisi, Matt J. Kusner, Timo Betcke, Stanislas Pamela, and Marc Peter Deisenroth. Learning physical operators using neural operators, 2026. URL https://arxiv.org/abs/2602.23113.

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers, 2022. URL https://arxiv.org/abs/2111.13587.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: Auto-regressive denoising operator transformer for large-scale PDE pre-training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17616–17635. PMLR, 2024. URL https://proceedings.mlr.press/v235/hao24d.html.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs), 2016. URL https://arxiv.org/abs/1606.08415.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/84e1b1ec17bb11c57234e96433022a9a-Abstract-Conference.html.

Hamda Hmida, Hsiu-Wen Chang Joly, and Youssef Mesri. CompNO: A novel foundation model approach for solving partial differential equations. *Applied Sciences*, 16(2):972, 2026. doi: 10.3390/app16020972. URL https://www.mdpi.com/2076-3417/16/2/972.

Matthias Karlbauer, Timothy Praditia, Sebastian Otte, Sergey Oladyshkin, Wolfgang Nowak, and Martin V. Butz. Composing partial differential equations with physics-aware neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10773–10801. PMLR, 2022. URL https://proceedings.mlr.press/v162/karlbauer22a.html.

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021. doi: 10.1073/pnas.2101784118. URL https://doi.org/10.1073/pnas.2101784118.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021. URL https://arxiv.org/abs/2010.08895.

Qiang Liu, Felix Koehler, Benjamin Holzschuh, and Nils Thuerey. Tadpole: Autoencoders as foundation models for 3d PDEs with online learning, 2026. URL https://arxiv.org/abs/2605.15284.

Yining Luo, Yingfa Chen, and Zhen Zhang. CFDBench: A large-scale benchmark for machine learning methods in fluid dynamics, 2023. URL https://arxiv.org/abs/2310.05963.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d7cb9db5ade2db7814fbd01ee59f4c7b-Abstract-Conference.html.

Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Regaldo-Saint Blancard, Francois Rozet, Cristiana Diaconu, Lucas Meyer, Kaze W. K. Wong, Hadi Sotoudeh, Alberto Bietti, Irina Espejo, Rio Fear, Siavash Golkar, Tom Hehir, Keiya Hirashima, Geraud Krawezik, Francois Lanusse, Rudy Morel, Ruben Ohana, Liam Parker, Mariel Pettee, Jeff Shen, Kyunghyun Cho, Miles Cranmer, and Shirley Ho. Walrus: A cross-domain foundation model for continuum dynamics. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/62018.

Rudy Morel, Jiequn Han, and Edouard Oyallon. DISCO: Learning to DISCover an evolution operator for multi-physics-agnostic prediction. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 44750–44774. PMLR, 2025. URL https://proceedings.mlr.press/v267/morel25a.html.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning, 2021. URL https://arxiv.org/abs/2001.04385v4.

Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A. Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining codomain attention neural operators for solving multiphysics PDEs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/bc75fa9843a7905bbed9d83895a88f7f-Abstract-Conference.html.

Louis Serrano, Jiequn Han, Edouard Oyallon, Shirley Ho, and Rudy Morel. Test-time generalization for physics through neural operator splitting, 2026. URL https://arxiv.org/abs/2602.00884.

Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968. doi: 10.1137/0705041.

Dengdi Sun, Xiaoya Zhou, Xiao Wang, Hao Si, Wanli Lyu, Jin Tang, and Bin Luo. NESTOR: A nested MOE-based neural operator for large-scale PDE pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6147–6156, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Sun_NESTOR_A_Nested_MOE-based_Neural_Operator_for_Large-Scale_PDE_Pre-Training_CVPR_2026_paper.html.

Jingmin Sun, Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Towards a foundation model for partial differential equations: Multioperator learning and extrapolation. *Physical Review E*, 111 (3):035304, 2025. doi: 10.1103/PhysRevE.111.035304. URL https://journals.aps.org/pre/abstract/10.1103/PhysRevE.111.035304.

Li Sun, Hongbo Lv, Zhikai Jiang, Zhongtian Sun, Lanxu Yang, and Philip S. Yu. Origo: Interpretable multi-physics PDE foundation model through neural operator splitting. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://icml.cc/virtual/2026/poster/61760.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/0a9747136d411fb83f0cf81820d44afb-Abstract-Datasets_and_Benchmarks.html.

Alessandro Trenta, Andrea Cossu, and Davide Bacciu. ComPhy: Composing physical models with end-to-end alignment. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/10f34ee79b62627b7ebf6279d35ea480-Abstract-Conference.html.

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

Hong Wang, Haiyang Xin, Jie Wang, Xuanze Yang, Fei Zha, Huanshuo Dong, and Yan Jiang. Mixture-of-experts operator transformer for large-scale PDE pre-training. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/2d23a9991a6f64482bf395628e279f5f-Abstract-Conference.html.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision*, pp. 3–19, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Yuxin_Wu_Group_Normalization_ECCV_2018_paper.html.

Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. PDEformer: Towards a foundation model for one-dimensional partial differential equations, 2024. URL https://arxiv.org/abs/2402.12652.

Yuan Yin, Vincent Le Guen, Jérémie Dona, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124012, 2021. doi: 10.1088/1742-5468/ac3ae5.

Xiangxiong Zhang and Chi-Wang Shu. On positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes. *Journal of Computational Physics*, 229(23):8918–8934, 2010. doi: 10.1016/j.jcp.2010.08.016. URL https://doi.org/10.1016/j.jcp.2010.08.016.

Hang Zhou, Yuezhou Ma, Haixu Wu, Haowen Wang, and Mingsheng Long. Unisolver: PDE-conditional transformers towards universal neural PDE solvers. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79061–79088. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25r.html.

A Implementation and Additional Results

A.1 Backbone implementation

The patch encoding, temporal aggregation, and grouped Fourier/local blocks below build on DPOT and MoE-POT (Hao et al., 2024; Wang et al., 2025).

All inputs use ten saved frames at a common 128^2 analysis resolution. The deterministic feature map constructs six channels per scalar field: normalized value, two coordinates, boundary distance, geometry support, and field validity. Both scales use patch size 8, four adaptive Fourier neural operator (AFNO) channel groups, and 16 retained modes. The scale-dependent dimensions are listed in Table 6. The context decoder expands to full resolution before two 3×3 convolutions and a 1×1 typed readout. Per-field history means and standard deviations are computed over the ten frames and spatial grid, with standard deviations floored at 10^{-6}. The system-specific mixers receive these means and log-scales in addition to shared features and task embeddings; the statistics have no learned parameters. Coordinates span $[-1, 1]$; boundary distance is constant on periodic grids and the minimum coordinate distance to an edge otherwise. These are input-feature conventions, separate from the expert's numerical update program.

Table 6: Backbone and system-specific predictor dimensions. The neural template and numerical programs are common to both scales.

Component	HotPOT-T (17M active)	HotPOT-S (90M active)
Shared width / depth / FFN width	256 / 2 / 512	512 / 3 / 2048
System-specific width / depth / FFN width	512 / 5 / 2048	1024 / 7 / 3968
Decoded context channels	32	64
Typed-head hidden width	128	256
Task-feature channels / hidden width	16 / 64	32 / 128
Task-embedding hidden width	128	128

Input encoding and positional embedding. For a grid of $N_x \times N_y$ cells, flattening only the spatial axes gives the feature matrix $\mathbf{X}^j \in \mathbb{R}^{N_x N_y \times d_{\text{in}}}$ at history position $j \in \{1, \dots, T\}$, with $d_{\text{in}} = 6$. The implementation embeds these six features in an 11-slot tensor with five fixed zero slots before the patch convolution. This fixed embedding is included in f_{patch}; parameter counts include the stored convolution weights. A convolutional patch encoder f_{patch} maps non-overlapping $p \times p$ patches to d-dimensional tokens, where $p = 8$ and $d = d_{\text{enc}}$ in this encoder. With $P = (N_x/p)(N_y/p)$ and learned positional matrix $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{P \times d}$,

$$\mathbf{Z}^j = f_{\text{patch}}(\mathbf{X}^j) + \mathbf{E}_{\text{pos}}, \qquad \mathbf{Z}^j \in \mathbb{R}^{P \times d}. \tag{12}$$

The patch encoder consists of a strided convolution, GELU (Hendrycks & Gimpel, 2016), and a pointwise projection. Its parameters, including $\mathbf{E}_{\text{pos}}$, are part of $\boldsymbol{\theta}$ and are shared across history frames, scalar fields, and datasets. Spatial convolutions and FFTs reshape these matrices to the two-dimensional grid and flatten them back in the same order.

Temporal aggregation. With field, setting, and saved-time indices suppressed, let $\mathbf{z}_i^j = (\mathbf{Z}_{i,:}^j)^\top \in \mathbb{R}^d$ be the token vector at patch i. The matrix $\mathbf{Z}_i^{\text{hist}} \in \mathbb{R}^{T \times d}$ in Fig. 2 stacks $\{(\mathbf{z}_i^j)^\top\}_{j=1}^T$. A real-valued adaptation of Fourier-feature aggregation uses learned channel-mixing matrices $\mathbf{W}_j \in \mathbb{R}^{d \times d}$ and frequency vector $\boldsymbol{\kappa} \in \mathbb{R}^d_{>0}$. Let $\lambda_j = (j - T)/(T - 1)$ denote the normalized temporal lag, $\odot$ elementwise multiplication, and $\mathbf{v}_i^0 \in \mathbb{R}^d$ the aggregated token. The cosine acts componentwise:

$$\mathbf{v}_i^0 = \sum_{j=1}^{T} \mathbf{W}_j^\top \left[\mathbf{z}_i^j \odot \cos(\boldsymbol{\kappa} \lambda_j) \right]. \tag{13}$$

Let $\mathbf{V}^0 \in \mathbb{R}^{P \times d}$ denote the matrix formed by stacking $\{(\mathbf{v}_i^0)^\top\}_{i=1}^P$.

Fourier and local mixing blocks. Spatial processing uses the grouped spectral mixer inherited from the MoE-POT implementation, following the AFNO/Fourier operator-transformer lineage

(Guibas et al., 2022). The mixer applies a two-dimensional fast Fourier transform (FFT), splits feature channels into groups, transforms each group with a two-layer frequency-domain multilayer perceptron (MLP), and returns to the spatial grid by inverse FFT. The MLP parameters are shared over retained frequency locations; GELU acts separately on real and imaginary components, with no soft-thresholding. Let f_{fft} and f_{ifft} denote the spatial FFT and its inverse, and let $f_{\mathrm{freq},b}$ be the grouped MLP with unretained modes set to zero. For block index b and input $\mathbf{U} \in \mathbb{R}^{P\times d}$, including the inherited internal skip connection gives

$$f_{\mathrm{spec},b}(\mathbf{U}) = \mathbf{U} + f_{\mathrm{ifft}}\left(f_{\mathrm{freq},b}(f_{\mathrm{fft}}(\mathbf{U}))\right). \tag{14}$$

Each block combines this spectral path with a local convolutional function $f_{\mathrm{local},b}$ and a channel feed-forward function $f_{\mathrm{ffn},b}$. The functions $f_{\mathrm{gn},b,1}$ and $f_{\mathrm{gn},b,2}$ are the block's two group normalizations (Wu & He, 2018). Let $\mathbf{V}^b$, $\mathbf{U}^b$, $\widetilde{\mathbf{V}}^b$, and $\mathbf{V}^{b+1} \in \mathbb{R}^{P\times d}$ denote the block input, normalized input, mixed features, and output, respectively:

$$\begin{aligned}
\mathbf{U}^b &= f_{\mathrm{gn},b,1}(\mathbf{V}^b),\\
\widetilde{\mathbf{V}}^b &= \mathbf{V}^b + f_{\mathrm{spec},b}(\mathbf{U}^b) + f_{\mathrm{local},b}(\mathbf{U}^b),\\
\mathbf{V}^{b+1} &= \widetilde{\mathbf{V}}^b + f_{\mathrm{ffn},b}\Big(f_{\mathrm{gn},b,2}(\widetilde{\mathbf{V}}^b)\Big).
\end{aligned} \tag{15}$$

All spatial feature matrices have shape $P \times d$, where d is the width of the shared or system-specific block in question. The local path uses a depthwise 3×3 convolution, GELU, and a 1×1 convolution; the FFN uses two 1×1 convolutions with GELU. After the shared blocks and a final normalization, the encoder produces $\mathbf{Z}_{r,c}$; their parameters belong to $\boldsymbol{\theta}$. The corresponding system-specific blocks use $\boldsymbol{\phi}_{\tau(r)}$.

A.2 System-specific numerical update programs

Each expert instantiates Eq. (5) by pairing a system-specific neural predictor with a numerical update program. Field-wise history encoding, system-specific state evolution, and spatial context decoding run once per saved interval. The program then constructs query states and reuses the predictor's readout. All predicted quantities are learned through next-state supervision; they require no separate rate, action, or increment labels. The programs below detail the five alternatives illustrated in Fig. 2. We suppress the setting index r below; N denotes the grid-cell count and a superscript $+$ the updated state. Vectors collect spatial entries; multi-channel arrays are written as matrices when channels form columns. The all-ones vector is $\mathbf{1}_N \in \mathbb{R}^N$, and $\langle \mathbf{a} \rangle = N^{-1}\mathbf{1}_N^\top \mathbf{a}$ is the spatial mean of a field vector $\mathbf{a}$.

A.2.1 NS: transport-rate integration

The neural transport-rate query in Eq. (4) uses stage vorticity, history context, and deterministic Biot–Savart transport and forcing features. Let $f_{\mathrm{rate}} : \mathbb{R}^N \to \mathbb{R}^N$ denote this query with the decoded history context and physical coefficients held fixed. Using the diffused and midpoint states $\boldsymbol{\omega}_a, \boldsymbol{\omega}_m$ from Eq. (6), its two outputs are

$$\widehat{\mathbf{r}}_1 = f_{\mathrm{rate}}(\boldsymbol{\omega}_a), \qquad \widehat{\mathbf{r}}_2 = f_{\mathrm{rate}}(\boldsymbol{\omega}_m). \tag{16}$$

The program adds forcing to each rate and uses the explicit midpoint update between two diffusion half-steps, forming a Strang split (Strang, 1968). The same decoded context and readout weights are reused at both queries and throughout the saved interval.

For the diffusion operator $f^s_{\mathrm{diff},\nu}$, let $\delta t_{\mathrm{CN}} > 0$ be a Crank–Nicolson time increment, s the diffusion duration, and $m = s/\delta t_{\mathrm{CN}} \in \mathbb{N}$ its number of increments. Let f_{fft} and f_{ifft} be the discrete Fourier transform on the periodic grid and its inverse. For Fourier mode ℓ, let $\lambda_\ell \geq 0$ be the eigenvalue of the negative Laplacian. With viscosity ν, define the mode multiplier b_ℓ and the diffusion update of a vorticity vector $\boldsymbol{\omega} \in \mathbb{R}^N$ by

$$b_\ell = \frac{1 - \delta t_{\mathrm{CN}}\nu\lambda_\ell/2}{1 + \delta t_{\mathrm{CN}}\nu\lambda_\ell/2}, \qquad f^s_{\mathrm{diff},\nu}(\boldsymbol{\omega}) = f_{\mathrm{ifft}}(\{b_\ell^m (f_{\mathrm{fft}}(\boldsymbol{\omega}))_\ell\}_\ell). \tag{17}$$

This composes m Crank–Nicolson diffusion increments (Crank & Nicolson, 1947). The implementation uses a positive multiplier, with $\delta t_{\mathrm{CN}}\nu\lambda_\ell/2 < 1$, and evaluates its power in logarithmic form. Each half-step takes $s = \delta t/2$; composing the split substeps covers the saved interval Δt_r.

A.2.2 CNS: CONSERVATIVE ACTION AND ADMISSIBLE UPDATE

At cell i, let $u_{x,i}, u_{y,i}$ denote velocity components, ρ_i density, and p_i pressure. Let $\gamma > 1$ be the specific-heat ratio; the total energy density is $E_i = p_i/(\gamma - 1) + \rho_i(u_{x,i}^2 + u_{y,i}^2)/2$. The conservative vector is $\mathbf{c}_i = (\rho_i, \rho_i u_{x,i}, \rho_i u_{y,i}, E_i)^\top \in \mathbb{R}^4$. The full conservative state is the matrix $\mathbf{Q} \in \mathbb{R}^{N\times 4}$ with row $\mathbf{c}_i^\top$. One predictor query supplies a normalized action, reshaped as $\mathbf{A} \in \mathbb{R}^{N\times 4}$. Its channel means are $\bar{\mathbf{a}} = N^{-1}\mathbf{A}^\top \mathbf{1}_N \in \mathbb{R}^4$. Let $\epsilon_s > 0$ be a numerical scale floor, and let $f_{\text{scale}} : \mathbb{R}^{N\times 4} \to \mathbb{R}^4$ return the per-channel spatial root mean square (RMS) of centered conservative fields, floored at ϵ_s; diag places these scales on a matrix diagonal. Denote the conservative increment by $\Delta\mathbf{Q}$ and its candidate state by $\mathbf{Q}^\star$. The program constructs

$$\Delta\mathbf{Q} = (\mathbf{A} - \mathbf{1}_N \bar{\mathbf{a}}^\top)\,\mathrm{diag}(f_{\text{scale}}(\mathbf{Q})), \qquad \mathbf{Q}^\star = \mathbf{Q} + \Delta\mathbf{Q}. \tag{18}$$

Multiplication by the diagonal matrix scales each channel uniformly over the grid. The program restores the candidate conservative means to those of $\mathbf{Q}$. Let $\rho_{\min}, p_{\min} > 0$ be density and pressure floors, and let $\alpha_{\text{CNS}} \in [0, 1]$ be one global coefficient ensuring these strict bounds in the updated state $\mathbf{Q}^+$. The segment update is

$$\mathbf{Q}^+ = \mathbf{Q} + \alpha_{\text{CNS}}(\mathbf{Q}^\star - \mathbf{Q}), \qquad \alpha_{\text{CNS}} \in [0, 1]. \tag{19}$$

For an admissible starting state, an admissible candidate uses $\alpha_{\text{CNS}} = 1$; otherwise a prescribed number B_{bis} of bisections determines an admissible lower endpoint of the segment. Boolean tests select the segment, and autograd differentiates its executed arithmetic. The result is converted back to primitive variables. The preserved quantities are the global totals of the four conservative channels on the uniform grid. Proposition 2 establishes the admissibility and conservation properties.

A.2.3 DR: TWO-QUERY PROPOSAL REFINEMENT

The sum of two predictor output groups gives a finite-interval concentration proposal $\boldsymbol{\delta}_1 \in \mathbb{R}^{2N}$ at the current state $\mathbf{q}^n$; the two concentration fields are concatenated in the same order as $\mathbf{q}^n$. A second query at an intermediate state gives $\boldsymbol{\delta}_2$; the program averages the two proposals as in Eq. (9). Both queries use the same context and readout weights. These outputs are concentration proposals learned through the resulting next-state error; the fixed operation is the query-and-average refinement.

A.2.4 SWE: MASS-PRESERVING HEIGHT INCREMENT

Only height is observed, with current field vector $\mathbf{h}^n \in \mathbb{R}^N$ and components h_i^n. One predictor query directly predicts a height increment $\Delta\mathbf{h} \in \mathbb{R}^N$ in the observed field units. The program centers it and applies one coefficient $\alpha \in [0, 1]$ to the entire grid, as specified in Eq. (8). Let $h_{\min} > 0$ be a prescribed height floor and $\epsilon_d > 0$ a numerical denominator floor. For an initial height satisfying $h_i^n \geq h_{\min}$, the coefficient in Eq. (20) ensures $\widehat{h}_i^{n+1} \geq h_{\min}$ at every cell i. Since the increment has zero spatial mean and α is uniform, the update preserves mass in exact arithmetic. The role of the limiter is to shorten the complete increment, rather than clip individual cells. Proposition 1 gives the formula and proof.

A.2.5 CFDB: NEXT-VELOCITY ASSIGNMENT

One predictor query estimates the next two velocity fields, concatenated as a vector in $\mathbb{R}^{2N}$. The program replaces the current velocity channels and copies auxiliary support channels unchanged, before the output mapping restores the evaluation resolution and channel layout. Within CFDB, spectral/local layers are shared across geometry variants, while pointwise FFNs and velocity heads are geometry-specific.

A.3 PROPERTIES OF THE NUMERICAL UPDATES

We establish the update properties used by the SWE and CNS numerical programs on a uniform grid. All arrays have finitely many finite-valued entries, arithmetic is exact, and each admissibility factor is a scalar shared by all spatial cells of one sample.

Proposition 1 (Mass and height-floor preservation)**.** *On a grid with N cells and prescribed height floor $h_{\min} > 0$, let $\mathbf{h} \in \mathbb{R}^N$ satisfy $h_i \geq h_{\min}$, and let $\mathbf{d} \in \mathbb{R}^N$ satisfy $\sum_i d_i = 0$. Given a*

denominator floor $\epsilon_d > 0$, denote the cellwise step bounds by r_i, their clipped minimum by α, and the updated height by $\mathbf{h}^+$. Define

$$r_i = \begin{cases} (h_i - h_{\min})/\max(-d_i, \epsilon_d), & d_i < 0, \\ 1, & d_i \geq 0, \end{cases} \qquad \alpha = \min(1, \min_i r_i), \qquad \mathbf{h}^+ = \mathbf{h} + \alpha \mathbf{d}. \tag{20}$$

Then $0 \leq \alpha \leq 1$, $h_i^+ \geq h_{\min}$ for every cell, and $\sum_i h_i^+ = \sum_i h_i$.

Proof. Every ratio is nonnegative. For $d_i \geq 0$, the update cannot decrease h_i. For $d_i < 0$, the denominator in Eq. (20) is at least $-d_i$, so $\alpha(-d_i) \leq h_i - h_{\min}$, which gives the height bound. Since the same α multiplies every cell, $\sum_i h_i^+ = \sum_i h_i + \alpha \sum_i d_i = \sum_i h_i$. If $\mathbf{d} = \mathbf{0}$, then $\alpha = 1$ and the state is unchanged. A cell at the floor with a negative increment instead gives $\alpha = 0$. □

Centering the directly predicted height increment gives $\mathbf{d}$. The denominator floor can shorten the permitted step for very small negative increments, but leaves both guarantees intact. The common scale is essential to this construction: cellwise clipping would generally alter the zero-sum increment and hence its mass balance.

Proposition 2 (CNS conservation and admissible segment)**.** *At one cell, let $\mathbf{m} = (\rho u_x, \rho u_y)^\top \in \mathbb{R}^2$ denote momentum, with components m_x, m_y, and $\mathbf{c} = (\rho, m_x, m_y, E)^\top \in \mathbb{R}^4$ the conservative state. For $\gamma > 1$ and positive floors $\rho_{\min}, p_{\min}$, let $p(\mathbf{c})$ be the pressure recovered from the conservative state and $\mathcal{K}$ its admissible set:*

$$\mathcal{K} = \left\{ \mathbf{c} \in \mathbb{R}^4 : \rho > \rho_{\min},\ p(\mathbf{c}) = (\gamma - 1)\left(E - \frac{\|\mathbf{m}\|_2^2}{2\rho}\right) > p_{\min} \right\}. \tag{21}$$

Let $\mathbf{c}_i$, $\mathbf{c}_i^\star$, and $\mathbf{c}_i^+$ be the column vectors corresponding to row i of $\mathbf{Q}$, $\mathbf{Q}^\star$, and $\mathbf{Q}^+$, respectively. Suppose every $\mathbf{c}_i$ lies in $\mathcal{K}$, and a finite candidate satisfies $\sum_i \mathbf{c}_i^\star = \sum_i \mathbf{c}_i$. The implemented segment search returns $\mathbf{Q}^+ = \mathbf{Q} + \alpha_{\mathrm{CNS}}(\mathbf{Q}^\star - \mathbf{Q})$ with every $\mathbf{c}_i^+ \in \mathcal{K}$ and $\sum_i \mathbf{c}_i^+ = \sum_i \mathbf{c}_i$. If $\mathbf{Q}^\star$ is admissible, $\alpha_{\mathrm{CNS}} = 1$. Otherwise, let $t \in [0, 1]$ denote a segment fraction and $\alpha_{\mathrm{CNS},\star}$ the supremum of admissible fractions:

$$\alpha_{\mathrm{CNS},\star} = \sup\{t \in [0, 1] : \mathbf{c}_i + t(\mathbf{c}_i^\star - \mathbf{c}_i) \in \mathcal{K} \text{ for every } i\}.$$

For a prescribed bisection count B_{bis}, the returned lower endpoint satisfies $0 \leq \alpha_{\mathrm{CNS},\star} - \alpha_{\mathrm{CNS}} \leq 2^{-B_{\mathrm{bis}}}$.

Proof. For two states with densities $\rho_0, \rho_1 > 0$ and momenta $\mathbf{m}_0, \mathbf{m}_1$, and a fraction $t \in [0, 1]$, weighted Cauchy–Schwarz gives

$$\frac{\|(1-t)\mathbf{m}_0 + t\mathbf{m}_1\|_2^2}{(1-t)\rho_0 + t\rho_1} \leq (1-t)\frac{\|\mathbf{m}_0\|_2^2}{\rho_0} + t\frac{\|\mathbf{m}_1\|_2^2}{\rho_1}.$$

Thus $p(\mathbf{c})$ is concave where $\rho > 0$, and the two strict inequalities in Eq. (21) define a convex set. Intersecting the cellwise admissible segments gives an interval containing zero. In the rejected-candidate branch, initialize lower and upper endpoints at 0 and 1. At every bisection, an accepted midpoint replaces the admissible lower endpoint; a rejected midpoint replaces the upper endpoint. Convexity keeps $\alpha_{\mathrm{CNS},\star}$ bracketed, and after B_{bis} iterations the bracket width is $2^{-B_{\mathrm{bis}}}$. The returned lower endpoint is admissible, including when it remains zero. For either branch,

$$\sum_i \mathbf{c}_i^+ = \sum_i \mathbf{c}_i + \alpha_{\mathrm{CNS}}\left(\sum_i \mathbf{c}_i^\star - \sum_i \mathbf{c}_i\right) = \sum_i \mathbf{c}_i.$$

□

Mean restoration constructs the conservative candidate required by the proposition. One segment coefficient then acts on all cells and all four conservative components, respecting the strict floors $\rho_{\min}, p_{\min}$ for a given specific-heat ratio $\gamma > 1$. Reapplying either program from its admissible output preserves the same constraints in exact arithmetic. The convex admissible set is also used in positivity-preserving finite-volume and discontinuous Galerkin schemes (Zhang & Shu, 2010). Here the segment acts on a whole-field candidate and preserves global totals.

A.4 Optimization and evaluation

Table 7: Main evaluation populations and fine-tuning schedule for the pre-training tasks at both model scales. Train is the trajectory count used per epoch; test is the fixed evaluated row count. $C_{\mathrm{pred},r} = |\mathcal{O}_r|$ counts scored physical channels, and L_{roll} counts model-feedback prediction steps after ten input frames. The learning rate is the maximum during E200 fine-tuning.

Setting	Native grid	Train	Test	$C_{\mathrm{pred},r}$	L_{roll}	Fine-tune LR
NS 10^{-5}	64^2	1,000	200	1	10	10^{-5}
NS 10^{-3}	64^2	1,000	200	1	20	10^{-3}
CNS	128^2	9,000	200	4	11	10^{-4}
SWE	128^2	900	60	1	91	10^{-3}
DR	128^2	900	60	2	91	10^{-3}
CFDB	64^2	9,000	1,000	2	10	10^{-5}

Joint pre-training. Both scales train for 1000 epochs, with 545 updates per epoch, using two devices and local batch 20 (global batch 40). Each trajectory contributes one ten-frame-to-one-frame training window per epoch. Over a ten-epoch cycle, sampling visits the initial window once and early, middle, and late non-initial windows three times each. Batches contain compatible tasks; the epoch plan retains the benchmark's per-dataset trajectory exposure. Adam uses $\beta_1 = \beta_2 = 0.9$ and weight decay 10^{-6}. OneCycleLR reaches 10^{-3} after 20% of training, with initial and final division factors both 10^4. NS (on its native grid) and DR receive norm-scaled Gaussian input noise with factor 5×10^{-4}; CNS, SWE, CFDB, and all targets remain clean. The loss is the one-step physical-channel relative error in Eq. (11). The 9000 CFDB training rows comprise 581 cavity, 8231 cylinder, and 188 tube examples; all three geometry paths therefore participate in pre-training.

Fine-tuning on the pre-training tasks. Starting from the fixed E1000 checkpoint, the shared encoder and selected system-specific predictor train for 200 complete epochs; other predictors remain frozen. Both scales use global batch 40, the same Adam settings and 20% warmup, and target-only phase-aware sampling. Maximum learning rates are specified per task in Table 7; input noise is disabled. Fine-tuning retains full per-epoch target exposure, giving 5,000 updates for each NS setting, 45,000 for CNS and CFDB, and 4,600 for SWE and DR. Learning rates are fixed from the T-scale held-out E200 rollout-validation diagnostics and reused at both scales. Final refits use the complete original training populations and the fixed E200 checkpoint.

Rollout metric and population. After the first ten frames, every predicted frame is fed back to the model. For each trajectory and physical channel, the released relative-L_2 metric flattens the spatial grid and the complete reported prediction window together. Channel errors are then averaged over scored channels, and trajectory errors equally over evaluated rows; the denominator includes 10^{-8}. Masks exclude padded cells and unscored channels. Thus an H10 entry measures the whole ten-frame prediction window. CFDB scores velocity channels, with geometry/support channels used only as inputs and state assembly. The fixed main population uses the first 1000 rows (zero-based indices 0–999) of the 1182-row packed CFDB test file, comprising 110 cavity and 890 cylinder rows. It is row-weighted; the separate condition study uses case-uniform aggregation. Main and ablation results use fixed final checkpoints. Reported runs use one training seed; few-shot and wave-transfer runs use seed 0.

Few-shot selection protocol. NS at $\nu = 10^{-4}$ supplies 9800 target-training trajectories, and CNS at $M = 1$ supplies 9000. The last 256 trajectories of each training source form a fixed validation set. The reported training subsets contain 16, 64, or 256 trajectories sampled from the remaining rows. Paired methods use identical training and validation indices. Training runs for 500 epochs with batch $\min(20, \text{shots})$, Adam with $\beta_1 = \beta_2 = 0.9$, weight decay 10^{-6}, and OneCycleLR with 20% warmup and division factors 10^4. NS HotPOT uses maximum learning rates 10^{-4}, 10^{-4}, and 10^{-5} at the three budgets, respectively, selected from the evaluated $\{10^{-4}, 10^{-5}\}$ candidates by rollout validation. MoE-POT uses 10^{-4} on both targets; CNS HotPOT also uses 10^{-4}. Each training trajectory supplies one phase-aware ten-frame-to-next-frame window per epoch. Every ten epochs, validation evaluates autoregressive rollout on all 256 validation trajectories over the target

test horizon. The checkpoint with lowest finite validation L2RE is used for testing. Both models update their shared and selected task parameters during adaptation. MoE-POT starts from its local E1000 pre-training checkpoint.

A.4.1 FEW-SHOT ROLLOUT EVALUATION

Each target–budget pair uses its validation-selected checkpoint, including normalization statistics. Evaluation starts from ten input frames and uses model feedback for 20 steps on NS and 11 on CNS, over 200 test trajectories per target. Masked per-channel space–time L2RE is averaged equally over trajectories.

A.5 PARAMETER STORAGE AND NUMERICAL QUERY COUNTS

Table 8 gives exact parameter counts at both scales; Table 9 compares activated and stored parameters across models.

Table 8: Exact parameter accounting at both scales. Storage includes all system-specific expert paths; an active path includes the shared encoder and the selected system or CFDB geometry path.

Quantity	HotPOT-T	HotPOT-S
All-PDE shared history encoder	2,011,648	11,901,952
Active NS/SWE path	16,770,577	89,584,673
Active DR path	16,903,188	90,112,036
Active CFDB geometry path	16,905,234	90,116,130
Active CNS path	17,178,004	91,185,956
Stored total	97,902,720	518,480,368

Table 9: Activated and stored parameters, in millions. MoE-POT counts are the published nominal values (Wang et al., 2025, Table 5); HotPOT ranges cover its active routes.

Model	Activated	Stored total
MoE-POT-T	17	30
HotPOT-T	16.77–17.18	97.90
MoE-POT-S	90	166
HotPOT-S	89.58–91.19	518.48
MoE-POT-M	288	489

Repeated use of the same predictor counts its weights once in the activated parameter total and contributes to runtime according to the number of calls.

Each setting updates the shared encoder and its selected family-specific predictor. The two NS viscosity settings use the same NS predictor. For CFDB, the encoder and geometry-shared layers train together with the selected geometry's pointwise FFNs and velocity head. The family selector is deterministic and has no learned router.

Numerical query counts. The history encoder, system-specific context blocks, and spatial decoder run once per saved interval. NS uses two readout calls per numerical substep; DR uses two per saved interval, and CNS, SWE, and each CFDB geometry path use one. Repeated calls reuse the same readout weights, so they increase computation without increasing the number of activated parameters. The present configuration has no matched wall-clock benchmark; parameter counts and query counts describe different aspects of its computational cost.

A.6 ADDITIONAL ABLATION DETAILS AND ANALYSES

History sharing and placement. Full uses one learned history encoder across PDE families, followed by private coupling and evolution. Learned private history copies the same encoder architecture into a separately trainable encoder for each family. The two NS settings retain the same family

assignment. Predictor architectures, numerical updates, task exposure, and active-path dimensions are retained; storage increases because the history weights are no longer shared. Late-shared evolution uses field-history tokenization before private coupling and moves the shared processing to the evolution stage. Late-private evolution uses the corresponding placement with family-private weights. Table 10 summarizes the four combinations.

Table 10: History/evolution placement and sharing controls. The mean equally weights six settings; all results use fixed E1000 checkpoints.

Model	Placement	Weights	Stored (M)	Mean L2RE
Full	History	Shared	97.90	0.0101
Learned private history	History	Private	105.95	0.0104
Late-shared evolution	Evolution	Shared	97.90	0.0286
Late-private evolution	Evolution	Private	105.95	0.0287

Direct-state and iterative controls. Direct abbreviates the Direct-state predictor control. Full and Direct retain the same encoder, system-specific mixers, context decoders, and readout-head architectures, with 97,902,720 stored parameters and identical activated counts. Their history features, statistics, task inputs, and masks match. Direct replaces the prediction–update interface with residual state prediction, retaining nominal readout-call counts. It therefore changes prediction quantities, numerical execution, and stage-state queries together. Generic iteration uses the same cached history context and shared readout weights to refine the NS state through state-dependent residual queries. Its intervention tests the NS numerical update against a learned iterative alternative. Other paths retain their configured update structures. All rows are trained under the same E1000 multi-PDE protocol.

Space–time visualization of two-dimensional tasks. The SWE and DR tasks have two spatial dimensions, x and y. The two DR concentration fields are denoted by u and v. Figure 3 stacks their two-dimensional error fields over forecast steps to form (x, y, n) space–time volumes, where n indexes the forecast step. The third rendering axis therefore represents time, not a third spatial dimension. We compare HotPOT with Late-shared evolution using fixed example 0 and all 91 forecast steps. Two complementary two-dimensional cuts accompany each volume pair: an x–y map at step 91 locates the final spatial error differences, and an x–time map at zero-based grid index $y = 64$ shows how the local differences evolve. These cuts expose local differences without occlusion by the volume rendering.

In these illustrated trajectories, the DR spatial cuts contain extended regions of lower HotPOT error, and the time cuts show differences persisting across many forecast steps. For SWE, the time cut localizes differences along the moving wave pattern. These local views complement the six-setting aggregate comparison in Table 2.

Full-population temporal errors. Figure 4 aggregates every trajectory in the six main-test populations. It connects the final aggregate scores in Tables 2 and 3 to the evolution of prediction error through time. The per-step metric differs from full-horizon L2RE, which forms one norm over the complete space–time window.

Figure 5 shows where Full and Generic iteration differ on four matched NS trajectories.

A.7 Few-shot transfer to the wave family

Data and initialization. Wave-Layer and Wave-Gauss are two datasets from one wave-equation family in the Poseidon benchmark (Herde et al., 2024). Neither belongs to either source checkpoint's six-dataset pre-training mixture. For each target, we use a fixed custom split of mutually disjoint original trajectories: 1,800 in the training pool, 100 for validation, and 200 for testing. Each original trajectory belongs to exactly one split; sampled temporal windows remain within that split. All methods use these same splits. The two targets are evaluated separately at 128×128 resolution with ten input frames and 11/6 forecast frames for Layer/Gauss. Nested subsets contain 16, 32, 64, or 128 training trajectories.

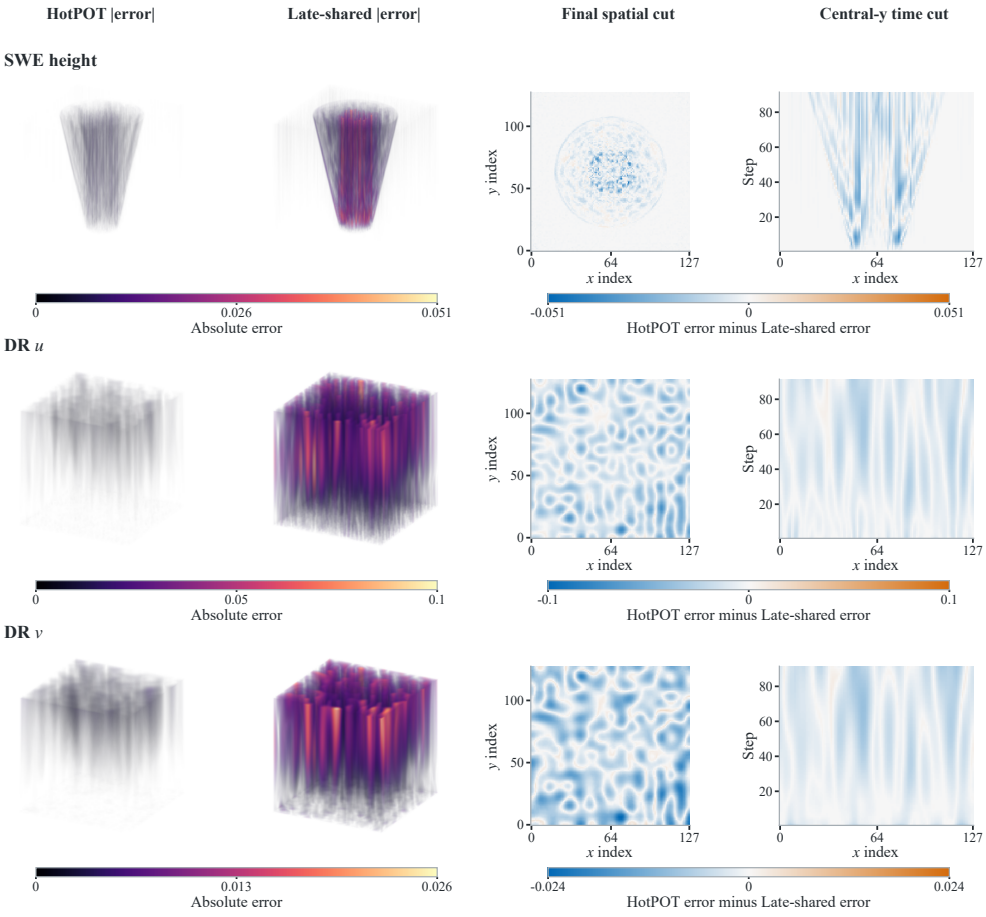

Figure 3: **Space–time errors and two-dimensional cuts for the sharing-location comparison.** Rows show SWE height and both DR fields for fixed example 0. The first two columns render absolute errors in two spatial dimensions and forecast time. The last two columns show HotPOT absolute error minus Late-shared evolution absolute error: an x–y slice at step 91 and an x–time slice at zero-based $y = 64$. Blue denotes lower error for HotPOT; orange denotes lower error for Late-shared evolution. Within each field, the error volumes share the camera, full-range color mapping, and opacity mapping; opacity increases with absolute error. The two cuts share a symmetric range determined by the full space–time error difference. Volume colorbars describe scalar values before compositing, whereas the cuts display local scalar differences directly.

HotPOT transfer initializes the 2.012M-parameter field-wise history encoder from the 17M E1000 source checkpoint. A scalar system-specific predictor is randomly initialized, and all 16.771M target-model parameters are trained. The scratch control has the same architecture and identical initial predictor weights, differing only in encoder initialization. Both use a common residual-prediction architecture that adds the predicted increment to the current displacement. Each model retains its native input normalization. MoE-POT loads its full local 1000-epoch pre-training checkpoint, updates all its trainable parameters, including the feature extractor and gating parameters.

Training and testing. All methods use the same target splits, sampled windows, training budget, optimizer, learning-rate schedule, and validation-selection rule. All runs use 500 epochs, batch $\min(20, \text{shots})$, Adam with $\beta_1 = \beta_2 = 0.9$ and weight decay 10^{-6}, and OneCycleLR with maximum 10^{-4}, 20% warmup, and both division factors 10^4. Each epoch samples one ten-frame-to-next-

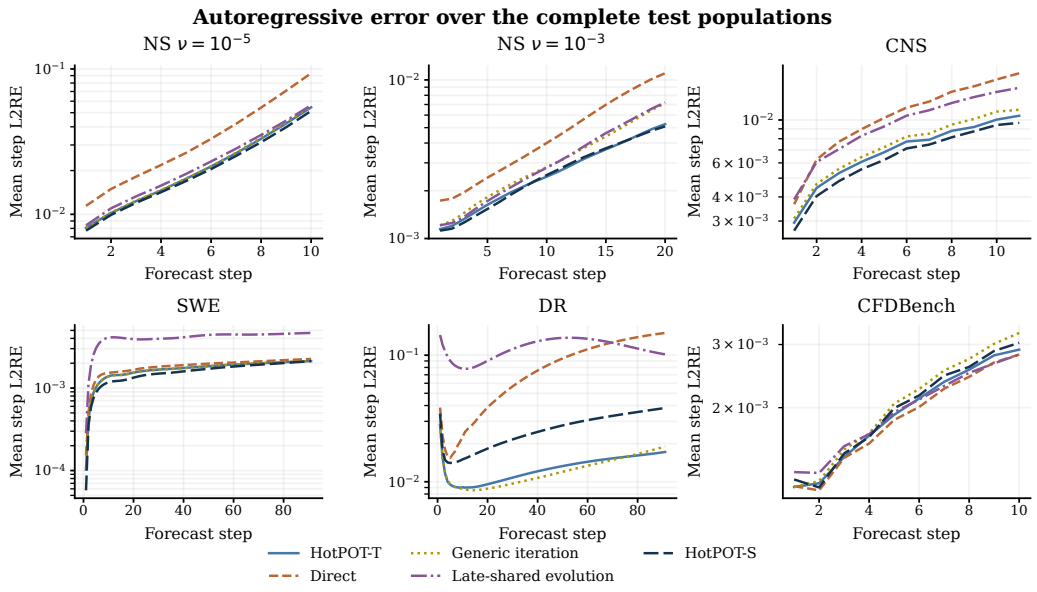

Figure 4: **Mean per-step errors across the complete test populations.** Each curve averages per-trajectory, scored-channel relative L_2 error at each forecast step. All models use fixed E1000 checkpoints and their own autoregressive feedback. CFDB uses the same row-weighted population as the main table. Vertical axes are logarithmic.

frame window per trajectory; the sampling indices and order are matched across arms. Validation evaluates the full rollout every ten epochs. The lowest validation error selects the checkpoint, with the earliest checkpoint breaking ties. The selected model is evaluated on the 200 held-out test trajectories.

The metric is the trajectory-uniform mean of per-step spatial L2RE, with denominator floor 10^{-8} and each model's own autoregressive feedback.

Table 11: Transfer to one unseen wave-equation family. Mean per-step L2RE on 200 custom held-out trajectories, averaged over 11/6 forecast frames for Layer/Gauss. All methods update their complete target models and select checkpoints by full-rollout validation within 500 epochs.

Target	Model	16	32	64	128
Wave-Layer	MoE-POT full fine-tuning	0.576	0.517	0.435	0.365
	HotPOT scratch	0.553	0.428	0.331	0.235
	HotPOT encoder transfer	**0.506**	**0.395**	**0.304**	**0.220**
Wave-Gauss	MoE-POT full fine-tuning	0.514	0.452	0.394	0.325
	HotPOT scratch	0.542	0.372	0.284	0.225
	HotPOT encoder transfer	**0.453**	**0.370**	**0.279**	**0.218**

Results. Encoder transfer lowers error relative to MoE-POT full fine-tuning by 12.1–39.8% on Wave-Layer and 11.9–32.9% on Wave-Gauss. It also improves on scratch training in all eight target–budget pairs, with reductions of 6.5–8.4% on Layer and 0.6–16.4% on Gauss. The comparison with scratch measures the effect of pre-trained encoder initialization under this matched target architecture and training protocol.

A.8 Physical diagnostics

For NS, relative enstrophy error compares the spatial mean squared vorticity with the reference at each forecast step and divides by the reference value, floored at 10^{-8}. For SWE and CNS,

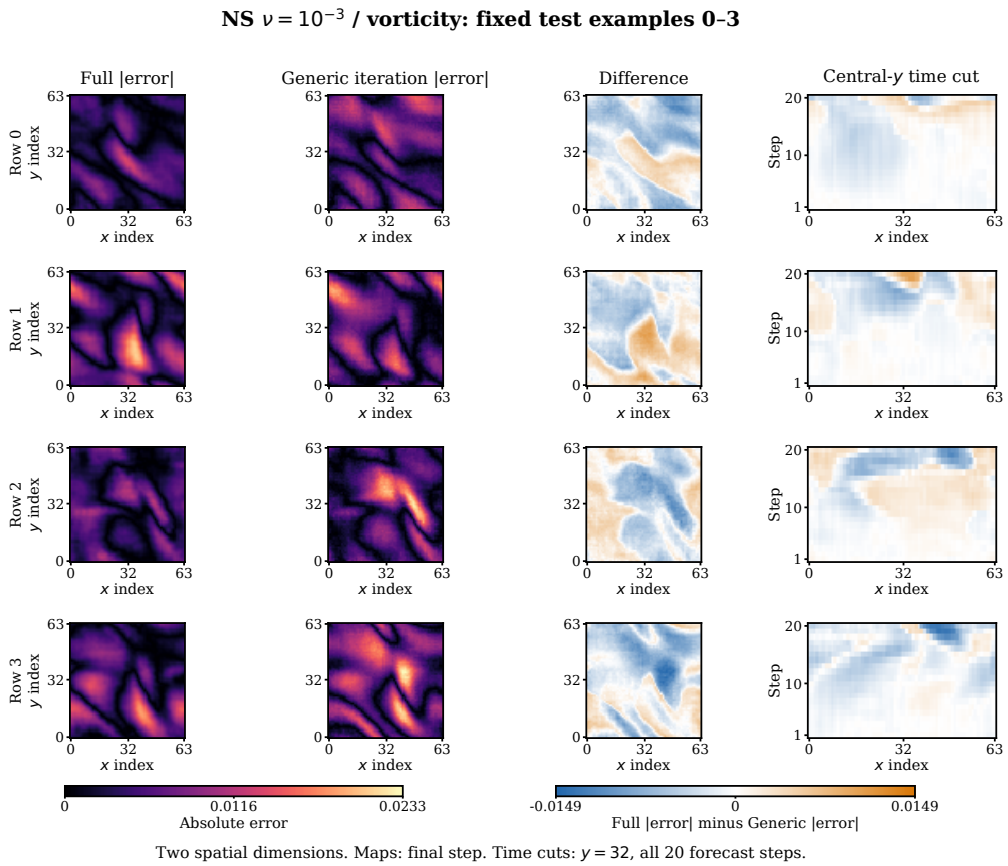

Figure 5: **NS numerical execution and generic iteration.** The first four test examples at $\nu = 10^{-3}$ use matched reference trajectories. The first two columns show final-step absolute errors; the third shows their difference. The fourth is an x–time cut at the central y index through the difference volume. Blue favors Full, orange favors Generic iteration. Error scales are shared across all four examples; difference scales are symmetric and use the complete forecast volumes. The task has two spatial dimensions; the rightmost vertical axis is time.

conservation drift measures the absolute change of a predicted spatial mean from the last observed state. SWE normalizes by the initial absolute mean height; CNS normalizes each conservative component by its initial spatial mean absolute value. Metrics average all forecast steps and then all test trajectories.

Table 12: Physical diagnostics over the complete test rollouts of fixed E1000 Full and Direct models. Values average all trajectories and forecast steps; lower is better.

System	Diagnostic	Full	Direct
NS 10^{-5}	Relative enstrophy error	$\mathbf{1.78 \times 10^{-3}}$	3.63×10^{-3}
NS 10^{-3}	Relative enstrophy error	$\mathbf{2.16 \times 10^{-3}}$	2.51×10^{-3}
CNS	Normalized mass drift	$\mathbf{7.63 \times 10^{-8}}$	2.04×10^{-4}
	Normalized energy drift	$\mathbf{8.15 \times 10^{-8}}$	2.06×10^{-4}
SWE	Relative mass drift	$\mathbf{1.19 \times 10^{-9}}$	9.31×10^{-5}

Full has lower mean enstrophy error on both NS settings and lower conservation drift on SWE and CNS than Direct in Table 12. Propositions 1 and 2 establish constraint preservation in exact arithmetic; the table measures the implemented floating-point rollouts. The CFDB program copies auxiliary support channels exactly.

Observed numerical execution. To distinguish mean correction from admissibility limiting, we observe the actual E1000 Full program on all 60 SWE trajectories over 91 steps and all 200 CNS trajectories over 11 steps. At each step, we record the uncentered neural proposal, the mean-corrected candidate, and the executed state along the model's own feedback trajectory.

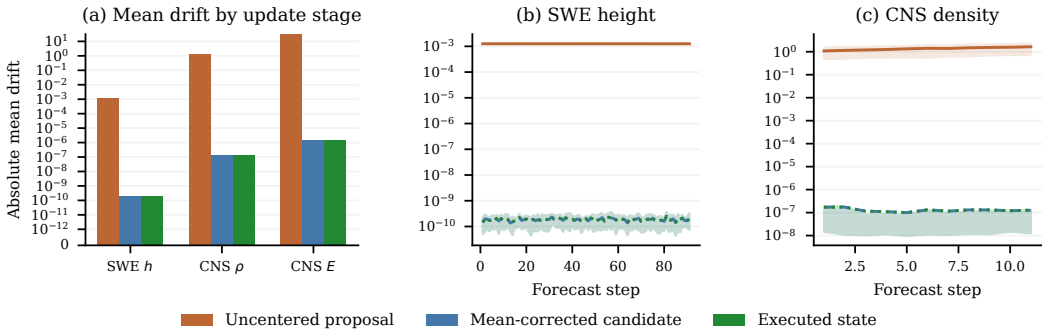

Figure 6: **Numerical execution along the fixed E1000 HotPOT rollouts.** (a) Mean absolute spatial-mean drift at each program stage, averaged over all recorded trajectory–step pairs for SWE height, CNS density, and CNS energy. (b,c) Per-step drift for SWE height and CNS density. Drift is the absolute change in spatial mean from the current state to the candidate next state, in released field units. Lines show arithmetic means and bands the 25th–75th percentiles across trajectories. All stages use the same current state on the model's own feedback trajectory.

Figure 6 shows the drift before and after mean correction. The limiter coefficient equals one at all 7,660 evaluated updates, so each mean-corrected candidate is executed unchanged. The minimum executed SWE height is 0.2519; the minimum CNS density and pressure are 0.05715 and 0.3181, respectively.

SWE reference compatibility. Across all 60 reference trajectories and their 91 forecast frames, float64 spatial reductions give the following compatibility measures. Relative to the last observed frame, the maximum absolute change in mean height is 1.20×10^{-9}. After shifting each forecast reference to that initial mean, both the initial and mean-shifted reference fields have minimum height above 0.2559, well above the 10^{-6} floor. The shifted references are compatible with the mass-preserving update in Appendix A.3. Let $\mathbf{h}^{\mathrm{ref},n} \in \mathbb{R}^N$ be the forecast reference field, $\bar{h}^{\mathrm{ref},n} = \langle \mathbf{h}^{\mathrm{ref},n} \rangle$ its spatial mean, and $\bar{h}^{\mathrm{obs}}$ the mean of the last observed input frame. The relative L_2 discrepancy associated with the initial mean, $\sqrt{N}|\bar{h}^{\mathrm{ref},n} - \bar{h}^{\mathrm{obs}}|/\|\mathbf{h}^{\mathrm{ref},n}\|_2$, averages 2.27×10^{-10} over trajectories and forecast steps.

A.9 Paired field visualizations

Figures 7 and 8 compare reference fields, model predictions, and absolute errors across six settings, including separate cavity and cylinder examples. Labels u_x, u_y denote velocity components in CNS and CFDB. All scored fields use shared color scales within each row. Both 17M-active models use fixed E1000 checkpoints, the same ten-frame history, and autoregressive feedback.

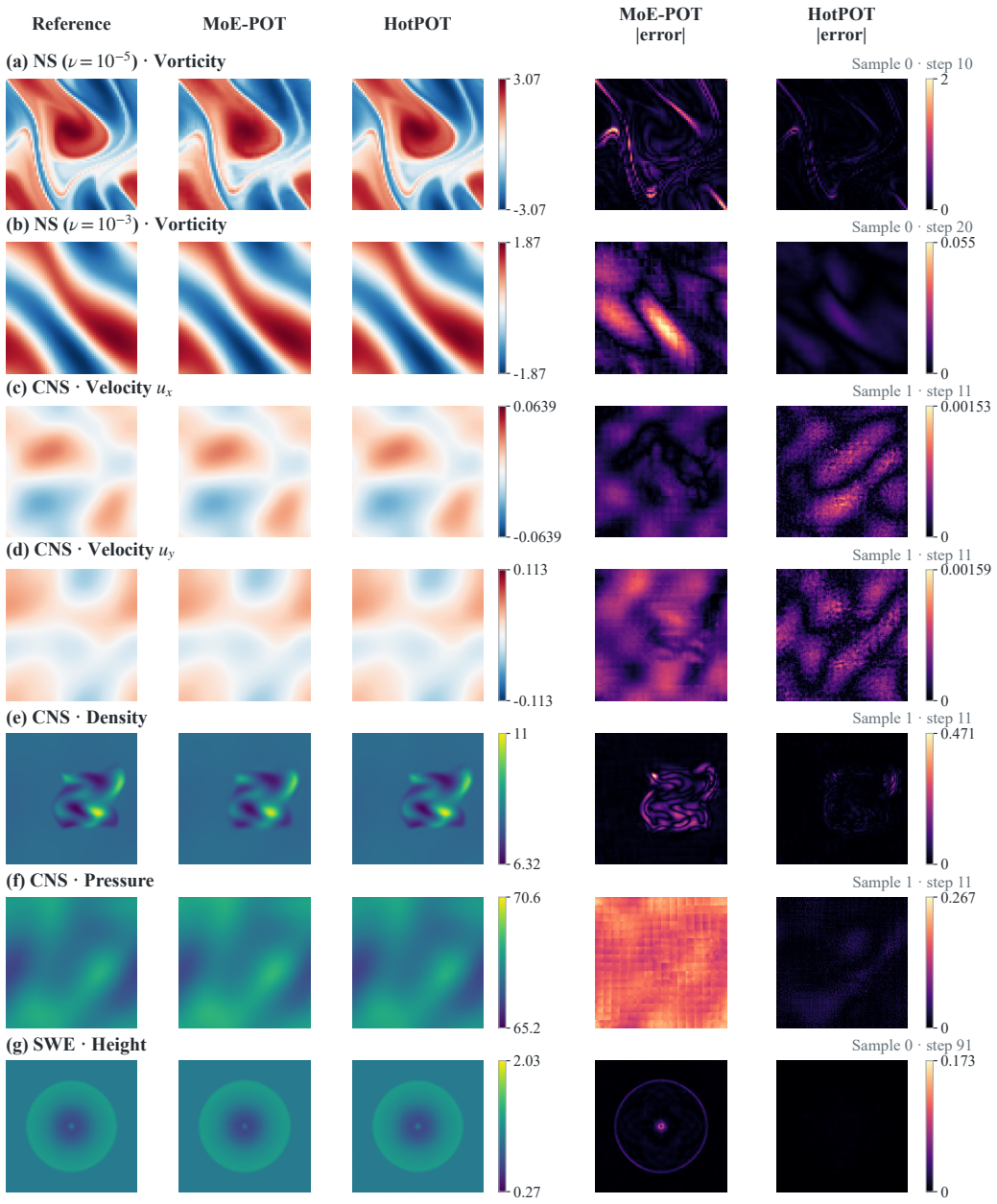

Figure 7: **Predictions and absolute errors: NS, CNS, and SWE.** All scored fields are shown at the final forecast step (NS/SWE: row 0; CNS: row 1). Reference and prediction fields share color limits within each row; error maps share a zero lower bound and the full-rollout maximum. Signed fields use zero-centered scales.

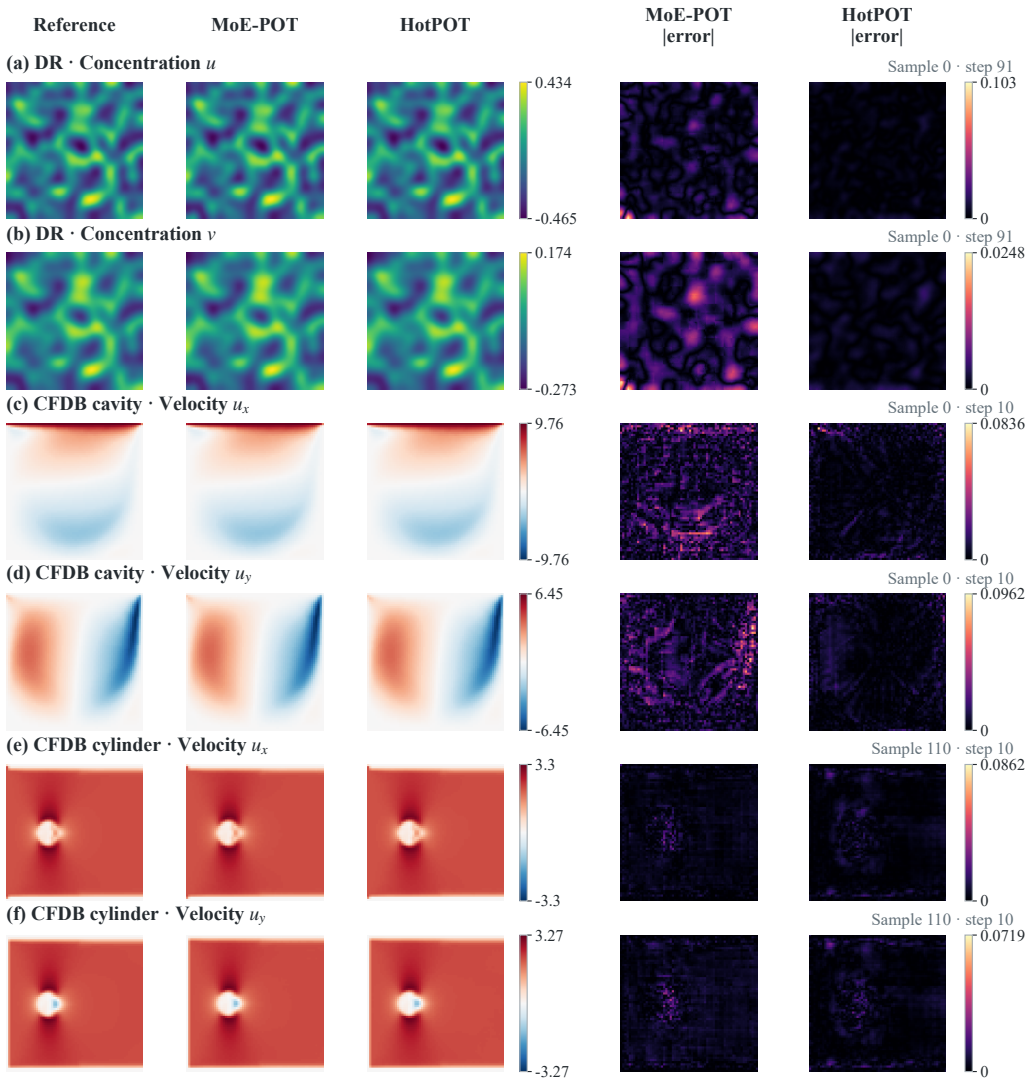

Figure 8: **Predictions and absolute errors: DR and CFDB.** All scored fields appear at the last forecast step (DR/cavity: row 0; cylinder: row 110). Within each row, reference and prediction fields share color limits; error maps share a zero lower bound and the full-rollout maximum.

Temporal error profiles. Figure 9 follows the same seven examples through every forecast step, complementing the final-frame maps with their temporal error profiles.

A.10 Space–time error structure

We examine the SWE and DR trajectories from Appendix A.9, retaining all 91 forecast steps and both DR fields. Fixing zero-based grid index $y = 64$ gives x–time cuts with forecast time on the vertical axis (Fig. 10). SWE shows outward-moving and returning waves; DR shows evolving concentration patterns. Color scales are shared within each field.

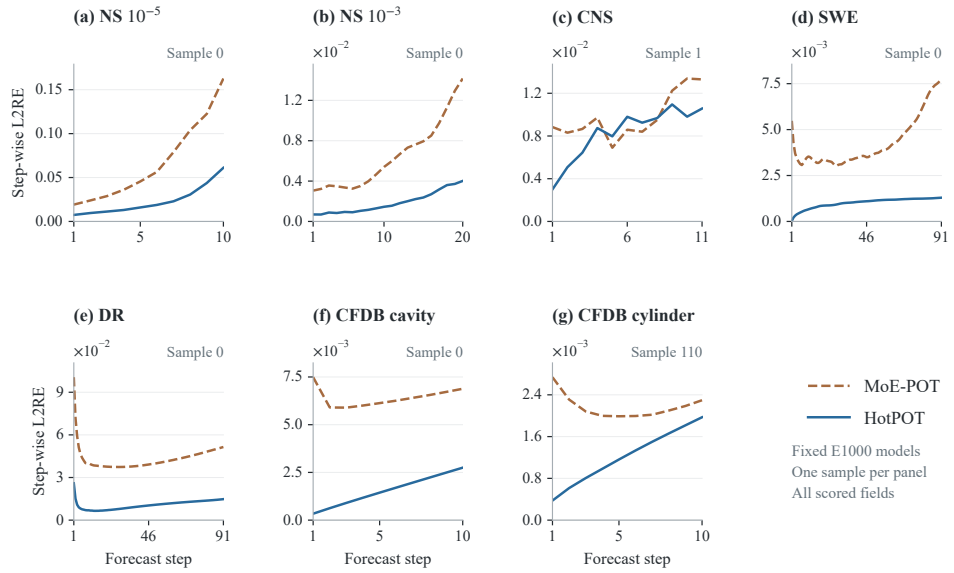

Figure 9: **Temporal errors for the seven illustrated trajectories.** Both E1000 models use autoregressive feedback from the same ten-frame histories. Each unsmoothed curve averages spatial L2RE over scored channels.

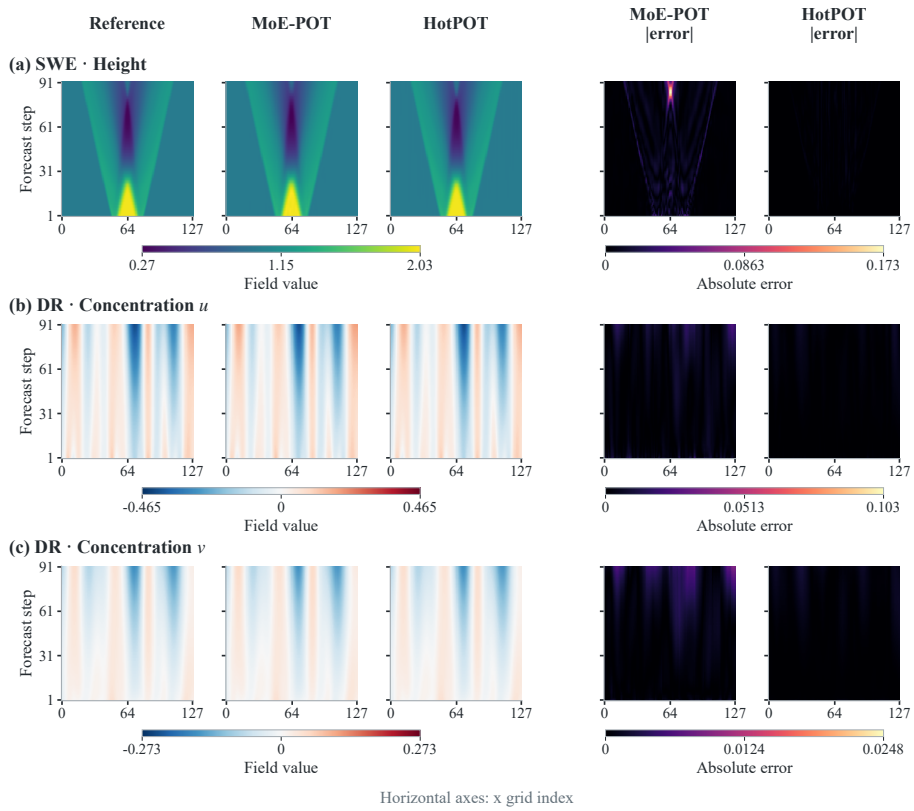

Figure 10: **Space–time slices of SWE height and DR u, v at zero-based $y = 64$.** Axes are x grid index and forecast step. Reference/prediction colors share full-array extrema within each row; error panels use zero to the full-array maximum. Both examples use trajectory row 0.