Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

SRPO: Deriving Neural Operators from Symmetry-Reduced Polynomial Interactions

Haichao Li

lihaichao@hrbeu.edu.cn

Harbin Engineering University

Xian He

Harbin Engineering University

Lufei Li

Harbin Engineering University

Fuzhen Pang

Harbin Engineering University

Yuan Du

Harbin Engineering University

Article

Keywords:

Posted Date: November 16th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11488718/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest.

SRPO: Deriving Neural Operators from Symmetry-Reduced Polynomial Interactions

Abstract

Neural operators typically learn mappings from predefined computational primitives. We instead derive the architecture from local PDE evolution, representing each update as a multivariate function and reducing redundancy before learning coefficients. This yields the Symmetry-Reduced Polynomial Operator (SRPO), combining relative polar coordinates, angular harmonics, low-rank radial/channel couplings, and multiscale first-, second-, and third-order interactions. Across five two-dimensional PDE benchmarks spanning reaction–diffusion, Navier–Stokes, shallow-water dynamics, and wave propagation in heterogeneous media, SRPO-P3 achieves the lowest held-out error on four tasks and ranks second on the fifth with 1.70 million parameters—about 2.5% of the full-spectrum Fourier Neural Operator (FNO) baseline. It reduces error by 38.0%–47.7% against the strongest non-SRPO baselines on Navier–Stokes and both wave benchmarks. Autoregressive rollouts and ablations indicate complementary contributions from multiscale information, angular harmonics, and higher-order interactions. These results show that PDE-informed structural reduction can yield compact operators with strong predictive accuracy.

Introduction

PDEs govern fluid flow, wave propagation, reaction–diffusion and free-surface dynamics, but repeated high-fidelity simulations across initial conditions, parameter fields or forecast horizons can be costly. Neural operators provide reusable surrogates for mappings between function spaces. DeepONet and the Fourier neural operator (FNO) established influential branch–trunk and global spectral formulations.[1–3] Subsequent work has explored convolutional, U-shaped and factorized spectral operators, transformer-based learning, learned geometric deformations and symmetry-aware Fourier operators.[4–11] Recent *Nature Machine Intelligence* studies have advanced Laplace-domain representations, neural integral equations, hybrid neural–numerical solvers, function-space architectures and operators for free-boundary or system-level problems.[12–17] This diverse literature leaves room for complementary principles for deriving operator structure.

Recent developments have addressed several limitations of standard spectral architectures. Methods for arbitrary or irregular domains include Fourier-based operators evaluated beyond regular grids, localized integral and differential kernels, reference-based geometry transfer and learned coordinate adaptation; the latter has also been explored in a recent preprint on Adaptive Coordinate

Transforms (ACT).[18–21] The Riesz neural operator and domain-decomposition approaches further emphasize local directional information and geometry generalization.[22,23] Other work targets the temporal and spectral demands of physical dynamics, including spherical operators, wavelet-diffusion models, explicit memory, continuous-time flow-matched prediction, physics-informed training, physics-agnostic pretraining, and data-efficient unsupervised pretraining with in-context learning.[24–30] These efforts are complementary to the architectural question addressed here: they improve geometry handling, temporal modelling, data efficiency or computational structure, whereas SRPO asks whether a compact interaction algebra can be derived directly from the local evolution map.

Evaluation has also broadened beyond isolated one-step errors. PDEBench and The Well provide standardized and diverse collections of physical simulations, while PDE-Refiner highlights the distinct challenge of maintaining accuracy during repeated autoregressive prediction.[31–33] In parallel, efficiency-oriented studies have developed decomposed spectral operators, locality constraints, neural-operator-based warm-start strategies, multigrid corrections, local operators for varying domains, Koopman-based formulations and full-frequency parameterizations; modular physical-operator learning provides another route to structuring the learned dynamics.[34–41] This literature motivates evaluation across multiple PDE families and, where suitable, under rollout—not only a single benchmark or a single-step metric.

To derive an operator architecture from the local PDE dynamics, we begin with the update at an individual target location. Given the discretized field at time t and any required physical coefficients or history, PDE evolution maps the input to the next frame. A neural operator approximates this mapping; applying it at every grid location produces the predicted frame. Trajectories, grid points and time steps provide repeated examples of the governing rule. We write the evolution and its approximation as:

$$\begin{aligned} u^{t+\Delta t}(x_0) &= F(\mathbf{u}^t, \mu; x_0) \\ \hat{u}^{t+\Delta t}(x_0) &= F_\theta(\mathbf{u}^t, \mu; x_0) \end{aligned} \tag{1}$$

For a fixed target location x_0, we collect the field values in its prescribed spatial neighbourhood, together with any required physical coefficients or history, into a finite input vector z. The value at x_0 in the next frame is the target output y. Thus, after spatial discretization, the local update is a multivariate map y = F(z), with each component of z representing a sampled neighbouring value or an included auxiliary input.

On a compact domain of these local input vectors z, multivariate polynomials are dense in the space of continuous functions. With sufficiently high degree and appropriately chosen coefficients, a

polynomial can therefore approximate the local evolution map y = F(z) to arbitrary accuracy. This provides a principled starting point: we represent the unknown map through a polynomial hierarchy:

$$F(\mathbf{z}) \approx P_0(\mathbf{z}) + P_1(\mathbf{z}) + P_2(\mathbf{z}) + P_3(\mathbf{z}) + \cdots, \tag{2}$$

Let $x_1, \ldots, x_N$ denote the selected input variables. The first three polynomial orders are:

$$\begin{aligned} P_1(\mathbf{x}) &= \sum_{i=1}^{N} a_i x_i, \\ P_2(\mathbf{x}) &= \sum_{i=1}^{N}\sum_{j=1}^{N} b_{ij} x_i x_j, \\ P_3(\mathbf{x}) &= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} c_{ijk} x_i x_j x_k. \end{aligned} \tag{3}$$

P_1 captures linear contributions, P_2 pairwise products, and P_3 three-way interactions; the zeroth-order term is handled by the learned pointwise branch. Explicit multiplicative and multi-body interactions have also been studied in polynomial neural networks and higher-order equivariant message passing, while bispectral methods use harmonic products to capture relationships that individual modes alone may miss.[42–44] An unrestricted expansion is expressive but quickly becomes impractical. For a scalar 128 × 128 field, treating all 16,384 grid values as independent variables gives N = 16,384; dense ordered quadratic and cubic coefficient tensors would then contain N^2 = 268,435,456 and N^3 = 4,398,046,511,104 entries. The growth is already severe on a much smaller 9 × 9 set of spatial samples (N = 81): ordered second-, third-, and fourth-order interactions contain 6,561, 531,441, and 43,046,721 tuples. Even after permutation-equivalent monomials are merged, a scalar polynomial still has 95,283 nonconstant monomials through degree three and 2,024,784 through degree four. These counts exclude additional channels and spatial scales. Crucially, the input variables are spatial samples with meaningful relative geometry, not arbitrary unrelated variables. This motivates the geometric reduction that follows.

Let x_0 denote the target location and x_i a neighboring point. The relative displacement is represented by a radius and an angle:

$$\boldsymbol{\xi}_i = \mathbf{x}_i - \mathbf{x}_0 = (r_i, \theta_i), \tag{4}$$

Because angular dependence is periodic, a spatial coefficient can be represented by a truncated Fourier series:

$$K(r, \theta) = \sum_{m=-M}^{M} K_m(r) e^{im\theta}. \tag{5}$$

This representation separates radial dependence from directional structure and replaces an unrestricted coefficient for every grid offset with a finite set of radial–harmonic components. Encoding geometric transformation laws in learned representations is also central to group-equivariant and steerable networks and recent equivariant neural operators.[45–47] The symmetry is imposed on the local interaction rule: rotated neighbourhood configurations share the same underlying coefficient structure, but the field values at those locations need not be equal. For example, in a 3 × 3 stencil centred at 5, the offset pair at positions 1–2 is a 180° rotation of the pair at positions 9–8, so the same local interaction rule is used for both configurations. The full PDE solution need not be rotationally symmetric. Boundaries, heterogeneous coefficients, anisotropic forcing and Cartesian discretization may still produce direction-dependent behaviour; non-zero modes are retained to represent this variation.

The same reduction determines the higher-order interactions. Under a common rotation, all neighbor angles shift by the same amount; the rotation-reduced quadratic coefficient therefore depends on the relative angle:

$$A(r_i, \theta_i, r_j, \theta_j) = A(r_i, r_j, \theta_i - \theta_j). \tag{6}$$

In harmonic coordinates, this relation couples opposite angular orders m and −m. For cubic interactions, the common-rotation phase cancels only for harmonic triplets satisfying the zero-sum selection rule:

$$m_1 + m_2 + m_3 = 0. \tag{7}$$

The P_2 and P_3 branches therefore do not enumerate every possible pair or triplet independently. The zero-sum rule identifies admissible cubic interactions, while the remaining radial and latent-channel couplings are compressed with low-rank factors. The resulting products of compact harmonic moments replace dense high-order coefficient tensors. P_1, P_2 and P_3 are consequently successive orders of the same polynomial-approximation principle after geometric reduction—not unrelated architectural heuristics. Together, these steps define SRPO-P3, whose architecture is summarized in Fig. 1.

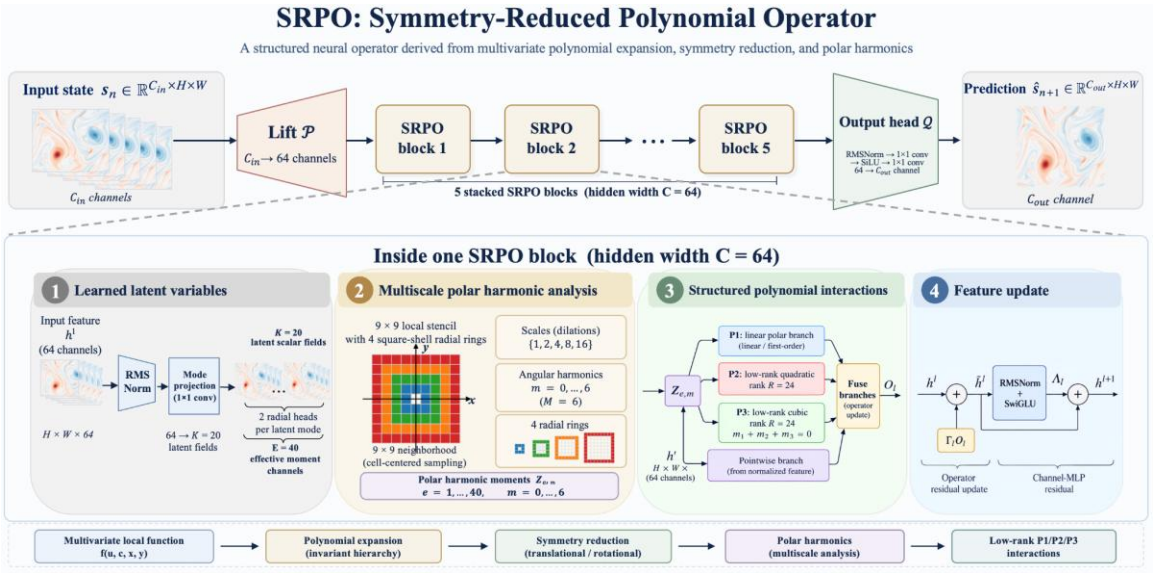


Figure 1 | SRPO-P3 architecture and derivation path. The input is lifted to a hidden representation and propagated through five SRPO blocks. Each block projects hidden features to learned latent scalar fields, constructs multiscale polar-harmonic moments, applies first-, second- and third-order interactions alongside a pointwise branch, and updates the hidden state through residual operator and channel-wise nonlinear paths. The bottom row summarizes the progression from local multivariate approximation to symmetry-reduced, low-rank interactions.

The default SRPO-P3 block projects the 64-channel hidden state to K = 20 latent scalar fields, then applies multiscale polar harmonics (M = 6; scales {1, 2, 4, 8, 16}) and rank-24 P2/P3 branches alongside a pointwise path. Five blocks yield approximately 1.70 million parameters. P4 extends the same selection rule to fourth order but is evaluated separately as a higher-order extension.

Results

Following the construction introduced above, we evaluate the default third-order model, SRPO-P3, on five PDE benchmarks using established benchmark protocols.[31,32] The Results are organized around three complementary questions: primary held-out prediction accuracy, autoregressive rollout behaviour, and the contributions of the model's structural components. We also evaluate SRPO-P4 as a higher-order extension, considering whether its additional interaction capacity justifies the increased model size and how its behaviour changes during rollout. The comparison uses competitive FNO-, CNO- and ACT-based baselines and is interpreted in the context of prior studies of neural-operator evaluation and model capacity.[48,49] The Navier–Stokes task follows the released Re = 5000 benchmark trajectories.[50]

Predictive accuracy across five PDE benchmarks

Table 1 reports each benchmark's primary held-out test metric; lower values indicate better performance within a column, while the metric definitions differ across datasets. SRPO-P3 achieves the lowest primary error on four of the five benchmarks and ranks second on shallow-water

dynamics. All models follow the corresponding benchmark's data split and input–output protocol, and checkpoints are selected using validation performance before held-out testing.

Table 1 | Primary held-out results across five PDE benchmarks. Best values are bold and second-best values are underlined; parameter counts are ranges across task-specific model instantiations.

Method	Parameters (M)	DiffReact RelL2	Navier–Stokes RelL2	SWE NRMSE	Wave-Gauss RelL2	Wave-Layer RelL2
Full-FNO	68.163–68.164	2.322×10^{-4}	0.101578	1.998×10^{-4}	0.059628	0.034517
ACT	68.191–68.192	2.371×10^{-4}	<u>0.054287</u>	1.705×10^{-4}	0.029211	0.015520
CNO	2.007–2.025	1.791×10^{-4}	0.160320	2.069×10^{-4}	0.044078	0.016492
CNO-Large	17.942–17.960	<u>1.545×10^{-4}</u>	0.126459	**1.367×10^{-4}**	<u>0.025042</u>	<u>0.009892</u>
SRPO-P3	1.699–1.701	**1.519×10^{-4}**	**0.033632**	<u>1.475×10^{-4}</u>	**0.014477**	**0.005172**

Full-FNO denotes the full-spectrum Fourier Neural Operator baseline, which uses spectral convolutions in Fourier space to model global spatial interactions. Compared with a conventional mode-truncated FNO, the full-spectrum configuration retains more Fourier modes and typically uses a larger parameter budget; this additional spectral capacity often leads to stronger performance on PDE tasks, although the gain depends on the task and implementation. Relative to Full-FNO, SRPO-P3 reduces the primary test error by 34.6% on DiffReact (1.519×10^{-4} vs 2.322×10^{-4}), 66.9% on Navier–Stokes at Re = 5000 (0.033632 vs 0.101578), 26.2% on SWE (1.475×10^{-4} vs 1.998×10^{-4}), 75.7% on Wave-Gauss (0.014477 vs 0.059628), and 85.0% on Wave-Layer (0.005172 vs 0.034517).

Against the strongest competing baseline, SRPO-P3 is 1.7% lower than CNO-Large on DiffReact (1.519×10^{-4} vs 1.545×10^{-4}), 38.0% lower than ACT on Navier–Stokes (0.033632 vs 0.054287), and 42.2% and 47.7% lower than CNO-Large on Wave-Gauss (0.014477 vs 0.025042) and Wave-Layer (0.005172 vs 0.009892), respectively. SWE is the exception: CNO-Large achieves the lowest NRMSE (1.367×10^{-4}), with SRPO-P3 ranking second at 1.475×10^{-4}. These accuracy gains also come with substantially fewer parameters: on DiffReact, SWE, Wave-Gauss and Wave-Layer, SRPO-P3 uses about 9.5% as many parameters as CNO-Large; on Navier–Stokes, it uses about 2.5% as many as ACT. Together, these results show that SRPO-P3 combines strong predictive accuracy with a compact parameterization across several PDE families. Extended qualitative comparisons and additional held-out examples are provided in Supplementary Note 5 (Supplementary Figures 1–7); the extended one-step quantitative results are summarized in Supplementary Table 8.

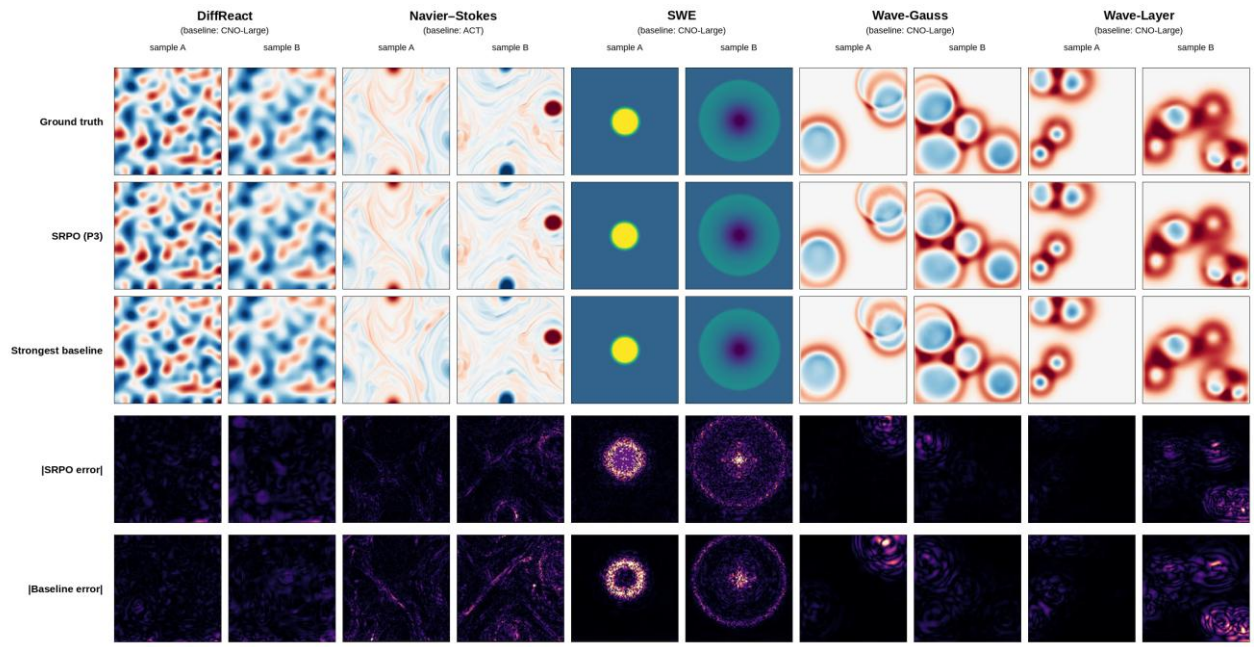


Figure 2 | Representative one-step predictions and absolute-error maps across all five PDE benchmarks. Each benchmark contributes two representative held-out examples. Within each benchmark-specific pair of columns, the five rows show the ground truth, the SRPO-P3 prediction, the strongest non-SRPO baseline prediction, the SRPO-P3 absolute error, and the baseline absolute error. The strongest baseline is ACT for Navier–Stokes and CNO-Large for DiffReact, SWE, Wave-Gauss and Wave-Layer.

Autoregressive rollouts reveal task-dependent stability

Autoregressive evaluation probes a different property from one-step testing: whether prediction errors remain controlled when a learned update is repeatedly composed. Importantly, all models reported here were trained with the benchmark's one-step supervised objective. We did not use a two-step or multi-step rollout-training objective. The rollout curves therefore evaluate the closed-loop behaviour of the validation-selected one-step checkpoint; they do not come from a separately trained multi-step model.

At evaluation, rollout begins from the prescribed ground-truth context. The model predicts the next state, and that prediction is fed back into the dataset-specific input history for the next update:

$$\hat{u}_{n+1} = F_\theta(H_n) \tag{8}$$

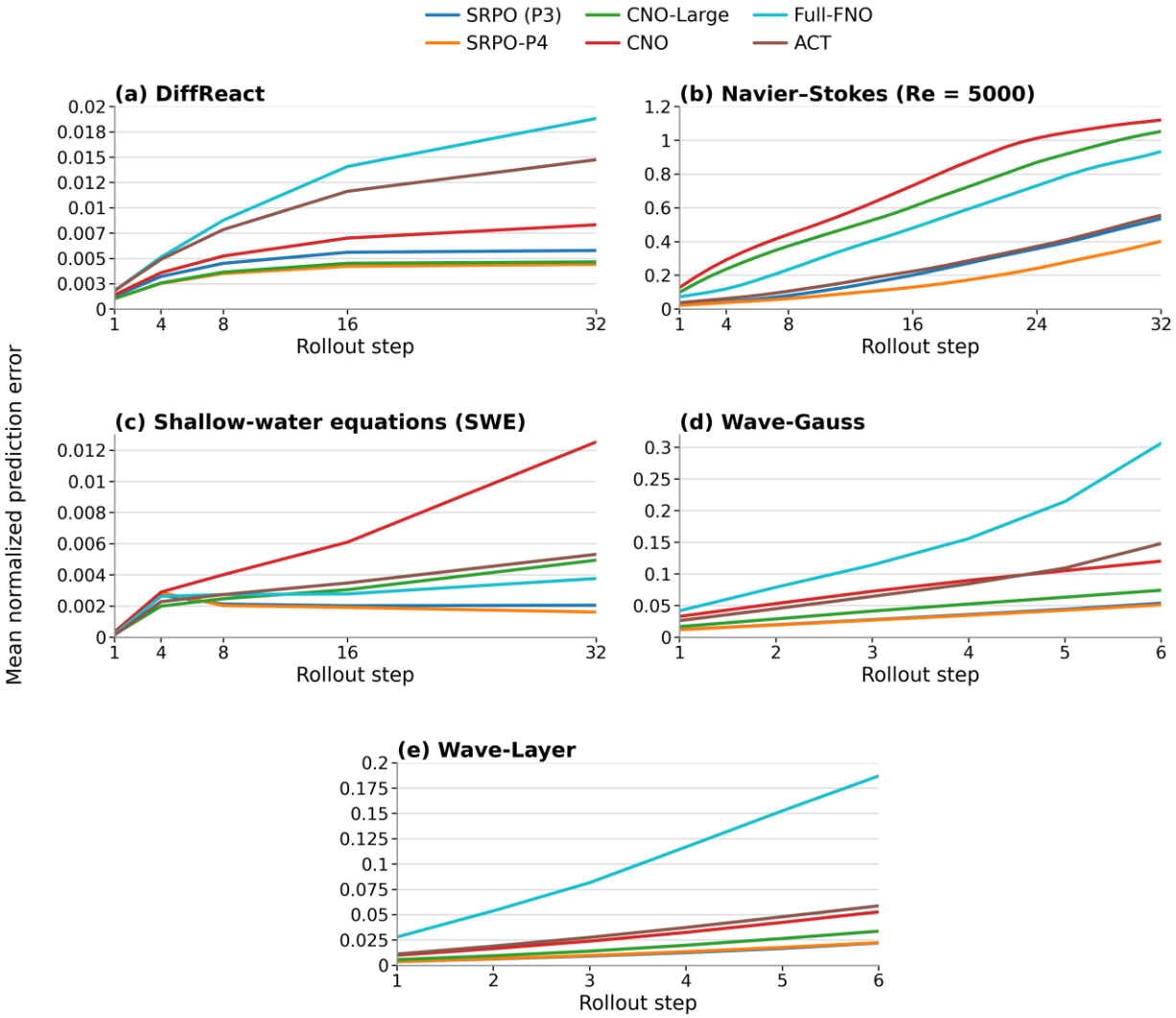
Here, H_n denotes the input history required by a given benchmark, including any prescribed conditioning. Its exact structure follows that benchmark's data protocol. After the initial ground-truth context, the history is updated using the model prediction rather than the future ground-truth state, so local approximation errors can accumulate over repeated updates.

The multi-step results show that long-horizon behaviour depends on the task (Fig. 3). On DiffReact, the final channel-mean NRMSE at t + 32 is 0.005794 for SRPO-P3, 0.004427 for SRPO-P4 and 0.004656 for CNO-Large, with P4 achieving the lowest error among these models. On

Wave-Gauss, the final mean physical RelL2 at t = 14 is 0.053781 for P3 and 0.051322 for P4; both are below CNO-Large (0.074340). On Wave-Layer, P3 reaches 0.022108, slightly below P4 at 0.022385, while CNO-Large reaches 0.033780. On Navier–Stokes, P4 remains the strongest SRPO variant across the plotted horizon and stays below the non-SRPO baselines, while on SWE both SRPO variants remain well below the competing baselines at longer horizons, with P4 achieving the lowest terminal NRMSE among the methods shown.

These results highlight that one-step accuracy and long-horizon performance provide complementary views of model behaviour. Figure 3 reports autoregressive rollout results for all five PDE benchmarks: DiffReact, Navier–Stokes, SWE, Wave-Gauss and Wave-Layer. Because the metrics and rollout horizons differ across datasets, rollout errors are interpreted within each benchmark rather than compared directly across benchmarks. Additional rollout visualizations and the corresponding quantitative summaries are provided in Supplementary Note 5 (Supplementary Figure 8 and Supplementary Tables 9–11).

Figure 3 | Autoregressive rollout error across five PDE benchmarks using validation-selected checkpoints trained with one-step supervision. (a) DiffReact; (b) Navier–Stokes (Re = 5000); (c) shallow-water equations (SWE); (d) Wave-Gauss; and (e) Wave-Layer. All panels use the common display labels "Rollout step" and "Mean normalized prediction error".



Note: Models are trained with one-step supervision only; no multi-step rollout loss is used during training.

Ablations reveal complementary structural contributions

We examined the main structural choices in SRPO-P3 through controlled Wave-Gauss ablations: explicit polynomial interaction order, nonzero angular harmonics, and multiscale spatial aggregation. All values in Table 2 are from the seed-2026 held-out evaluation. Pair RelL2 measures one-step, teacher-forced prediction error; rollout RelL2 measures the autoregressive error at t = 14. Lower values are better.

Table 2 | Wave-Gauss structural ablations. Parameter reduction is relative to the full SRPO-P3 model.

Variant	Parameters	Pair RelL2	Rollout t = 14 RelL2	Parameter reduction
Full SRPO-P3	1.699 M	0.014477	0.053781	—
P1 + P2 (no P3)	0.595 M	0.018039	0.103211	64.98%
P1 only (no P2/P3)	0.353 M	0.023693	0.211339	79.22%
M = 0, full order	0.254 M	0.032898	0.129870	85.05%
Single scale, s = 1	1.676 M	0.028317	3.139862	1.32%

Note: The P1-only variant removes the explicit P2/P3 interaction algebra but retains the nonlinear MLP path; the complete network is therefore not globally linear. The Single-scale variant reaches a final t = 14 rollout error of 3.139862.

Removing the cubic branch (P1 + P2) increases pair error by 24.6% and the final rollout error by 91.9%, while reducing parameters by about 65%. Removing the quadratic branch as well further increases pair and rollout errors to 0.023693 and 0.211339. Setting M = 0, which removes nonzero angular harmonics while retaining the full interaction-order hierarchy, yields 2.27 times the full-model pair error and 2.41 times its rollout error. The clearest evidence for multiscale aggregation comes from the single-scale variant: it retains 98.7% of the parameters but nearly doubles pair error, and its t = 14 rollout error rises to 3.139862, about 58.38 times the full-P3 value. Together, these results indicate that higher-order interactions, angular information and cross-scale spatial access make complementary contributions to the full SRPO-P3 model. Further ablation visualizations and the detailed sensitivity protocol are provided in Supplementary Note 6 (Supplementary Figures 9–13 and Supplementary Tables 12–16).

As a complementary higher-order extension—not an additional ablation—SRPO-P4 adds an explicit quartic interaction branch. Relative to P3, its one-step test error is lower on all five benchmarks, with reductions of 13.8% on DiffReact, 10.2% on Navier–Stokes, 6.7% on SWE, 0.7% on Wave-Gauss and 3.6% on Wave-Layer. This extension increases the parameter count from about 1.70 M to 3.24 M (approximately 1.91 times as many parameters) and incurs substantially greater training cost in the current implementation. Against the non-SRPO baselines, SRPO-P3 already achieves the best primary held-out result on four of five tasks with a substantially smaller parameter budget, so we use P3 as the default model for the main comparisons. P4 is retained as a higher-order

extension to examine potential accuracy gains, which must be weighed against its additional training burden and task-dependent rollout behaviour. The full P3–P4 single-step and final-horizon rollout comparisons are provided in Supplementary Tables 15 and 16.

Discussion

SRPO derives operator structure from the evolution map rather than selecting a predefined computational primitive. A local polynomial hierarchy is expressive, but unrestricted interaction tensors grow rapidly with neighbourhood size and order; SRPO uses relative geometry and symmetry-informed reductions before learning the remaining coefficients. P3 balances expressive capacity and implementation cost: it sets the highest explicit interaction order in each block, not the order of the entire network. Stacked nonlinear blocks can compose these interactions into more complex evolution.

SRPO uses relative polar coordinates to separate radial and angular dependence; angular harmonics represent directional variation, while the zero-sum rule selects higher-order coefficient combinations invariant to a common rotation. Low-rank factors compress the remaining radial and latent-channel couplings. This constrains local interaction coefficients, not the field itself: boundaries, heterogeneous media, anisotropy and grid discretization can still produce direction-dependent values. The contribution is deriving these reductions as an ordered construction from local geometry, rather than selecting modules independently.

The structural ablations provide empirical support for the complementary roles of these components. On Wave-Gauss, progressively introducing the explicit P2 and P3 branches improves both one-step prediction and autoregressive rollout. Removing non-zero angular harmonics degrades prediction, indicating that radial aggregation alone does not capture all relevant directional information. The single-scale variant is particularly revealing: despite having a parameter count close to that of the complete P3 model, it develops substantially larger errors during rollout. This contrast suggests that the spatial organization of the representation matters alongside its capacity. Taken together, the ablations support a division of labour in which multiscale radial aggregation provides access to spatial context, angular harmonics encode direction, and explicit higher-order interactions enrich the relationships among local variables.

The comparison uses competitive neural-operator baselines. FNO is a foundational spectral architecture; CNO targets robust PDE learning, and ACT adds learned coordinate adaptation. Full-FNO retains more spectral modes and typically a larger parameter budget than mode-truncated FNO,

making it a higher-capacity baseline that often performs better on PDE tasks. CNO-Large tests whether a wider baseline closes the gap to SRPO-P3.

These design choices are reflected in the five-benchmark evaluation. With approximately 1.70 million trainable parameters—the fewest among the compared models—SRPO-P3 achieves the best primary held-out result on four tasks and ranks second on shallow-water prediction. The clearest margins are on Navier–Stokes, where its test relative L2 error is 38.0% lower than ACT, and on Wave-Gauss and Wave-Layer, where its teacher-forced Pair RelL2 errors are 42.2% and 47.7% lower than CNO-Large, respectively. On DiffReact, SWE, Wave-Gauss and Wave-Layer, SRPO-P3 uses about 9.5% as many parameters as CNO-Large, the strongest non-SRPO accuracy baseline on those tasks; on Navier–Stokes, it uses about 2.5% as many as ACT. The advantage is not uniform: the DiffReact result is only slightly better than CNO-Large, and CNO-Large attains the lowest SWE NRMSE, with SRPO-P3 ranking second. The central result is therefore a strong accuracy–parameter balance across different PDE families, not universal dominance on every task or metric.

Rollouts add a distinct perspective because one-step errors and dynamical biases can accumulate differently when each prediction becomes the next input. The outcomes are task-dependent: P4 improves the final reported error over P3 on DiffReact and Wave-Gauss, whereas P3 remains slightly better on Wave-Layer. Thus, the one-step results do not imply uniform long-horizon dominance; evaluating both the learned update and its repeated composition is important for dynamical systems.

P4 reduces one-step error on all five datasets, with gains from 0.7% on Wave-Gauss to 13.8% on DiffReact, but does not consistently improve rollouts. It increases parameters from about 1.70 million to 3.24 million and requires substantially more training time. Because P3 already achieves the best primary held-out result on four of five tasks, it remains the default; P4 is a higher-order extension for studying when extra interaction capacity justifies its cost.

More broadly, SRPO suggests that neural-operator design can benefit from deriving a compact representation of the target mapping before scaling a generic computational primitive. The main benchmark comparison uses a single random seed (2026), so seed-to-seed variability and statistical significance are not quantified. Evaluation across broader PDE families and multiple seeds would help characterize robustness and range of applicability. Continued implementation work and controlled runtime profiling will also clarify how the structured interactions can be executed more efficiently, particularly for higher-order variants.

Methods

Design principle.

The Introduction presents the motivation and central symmetry-reduction argument and summarizes the SRPO-P3 architecture in Fig. 1. Here, we specify the corresponding finite-dimensional mapping, discrete harmonic kernels, low-rank factors and network implementation. SRPO is derived from the target evolution operator rather than assembled from a predefined computational primitive: the reverse design pass determines which interactions are represented, while the forward implementation computes them in a trainable network.

From physical evolution to a local polynomial hierarchy.

Let $\mathbf{s}_t$ denote the state required to predict the next state of a deterministic physical system, including the current field and, where necessary, historical frames, static coefficients or coordinate channels. We seek an approximation to the time-marching operator

$$\mathbf{s}_{t+1} = \mathcal{G}(\mathbf{s}_t; \boldsymbol{\mu}), \tag{9}$$

where $\boldsymbol{\mu}$ collects known physical parameters or static fields. After spatial discretization, the prediction at a reference location $\mathbf{x}_0$ can be expressed as a function of N local variables $\mathbf{z} = (z_1, \dots, z_N)$. For a sufficiently regular finite-dimensional mapping on a compact input domain, a multivariate polynomial provides a general approximation family. Omitting the constant term, which can be represented by biases in the pointwise transformations, we write the local mapping as

$$\mathcal{F}(\mathbf{z}) \approx P_1(\mathbf{z}) + P_2(\mathbf{z}) + P_3(\mathbf{z}). \tag{10}$$

The first three interaction orders are

$$P_1(\mathbf{z}) = \sum_{i=1}^{N} a_i \, z_i, \tag{11}$$

$$P_2(\mathbf{z}) = \sum_{i=1}^{N} \sum_{j=1}^{N} b_{ij} \, z_i z_j, \tag{12}$$

and

$$P_3(\mathbf{z}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} c_{ijk} \, z_i z_j z_k. \tag{13}$$

Here, P_1, P_2 and P_3 represent single-variable, pairwise and three-way interactions. This decomposition is expressive, but an unrestricted parameterization requires $O(N)$, $O(N^2)$ and $O(N^3)$ coefficients, respectively. Directly learning the coefficient tensors is therefore impractical for a spatial neighborhood. The key observation is that the z_i are not arbitrary variables: they are samples of a field at locations with a known spatial relationship. SRPO uses that geometry to reduce the coefficient degrees of freedom before learning the remaining interactions. The third-order truncation

is a practical choice, not a claim that every physical update is exactly cubic; composing several such blocks with nonlinear channel transformations produces effective interactions of higher order.

Relative geometry and polar-harmonic first-order interactions.

Let $\mathbf{x}_i$ be a neighboring grid point relative to $\mathbf{x}_0$. In two dimensions, its displacement is represented by a radius r_i and angle θ_i,

$$\mathbf{x}_i - \mathbf{x}_0 = r_i(\cos\theta_i, \sin\theta_i), \qquad r_i = \| \mathbf{x}_i - \mathbf{x}_0 \|_2. \tag{14}$$

This representation makes spatial weight sharing explicit: the interaction depends on relative geometry rather than on an independent absolute-grid index. For the first-order term, the coefficient a_i is replaced by a learnable kernel $K(r_i, \theta_i)$. Because angle is periodic, its dependence can be expanded in angular Fourier modes. Truncating the expansion at angular order M gives

$$K(r, \theta) = \sum_{m=-M}^{M} k_m(r) e^{im\theta}, \qquad Z_m(\mathbf{x}_0) = \sum_{i=1}^{N} k_m(r_i) e^{im\theta_i} z_i. \tag{15}$$

The scalar first-order contribution is a learned linear combination of the harmonic moments,

$$P_1(\mathbf{z}) = \sum_{m=-M}^{M} w_m Z_m(\mathbf{x}_0). \tag{16}$$

The Z_m are local polar-harmonic moments. The $m = 0$ component represents radial, orientation-independent aggregation, while non-zero orders retain directional variation. For real-valued fields, negative orders are represented by conjugacy, $Z_{-m} = \overline{Z_m}$, so the implementation only needs to store $m = 0, \dots, M$. In the network, this combination is implemented by a pointwise projection of the real-valued moment components.

This is a constraint on local coefficient sharing, not a claim of global rotational equivariance for the realized field. Rotated neighbourhoods share the corresponding coefficient structure, while boundaries, heterogeneous media, anisotropy and Cartesian discretization can still produce direction-dependent fields. Non-zero angular modes and pointwise channel mixing retain this directional and spatially heterogeneous information.

Symmetry reduction and low-rank higher-order interactions.

The harmonic moments transform simply under a common rotation of the local neighborhood by angle ϕ:

$$Z_m \mapsto e^{im\phi} Z_m. \tag{17}$$

The phase law identifies which harmonic products are invariant under a common rotation: for the scalar rotation-reduced coefficients used here, the retained harmonic orders must sum to zero. This

condition constrains the local interaction coefficients, not the sampled field values or global symmetry of a solution, and is used to construct the structured P2 and P3 branches.

For the quadratic term, write the geometric coefficient as $A(r_i, \theta_i, r_j, \theta_j)$. The rotation-reduced component is invariant to a common angular shift, so its angular dependence can be expressed through the relative angle $\Delta\theta = \theta_i - \theta_j$:

$$A(r_i, \theta_i, r_j, \theta_j) = \tilde{A}(r_i, r_j, \Delta\theta) = \sum_{m=-M}^{M} C_m(r_i, r_j) e^{im(\theta_i - \theta_j)}. \tag{18}$$

Symmetry removes one of the two absolute angular degrees of freedom, but C_m remains a function of two radii. SRPO compresses this residual dependence using a rank-R_2 separable factorization,

$$C_m(r_i, r_j) \approx \sum_{\alpha=1}^{R_2} f_{m,\alpha}(r_i) g_{m,\alpha}(r_j). \tag{19}$$

The projected moments are

$$U_{m,\alpha} = \sum_i f_{m,\alpha}(r_i) e^{im\theta_i} z_i, \qquad V_{m,\alpha} = \sum_j g_{m,\alpha}(r_j) e^{im\theta_j} z_j. \tag{20}$$

The quadratic interaction is then

$$P_2 = \mathcal{R}_2\left(\{U_{m,\alpha}\overline{V_{m,\alpha}}\}_{m,\alpha}\right), \tag{21}$$

The conjugate product follows from the relative-angle factor $e^{im(\theta_i - \theta_j)}$; it is not an independently chosen nonlinear operation.

For the cubic term, expand the angular dependence of the three-point coefficient in a three-dimensional Fourier basis. Under a common rotation, a term indexed by (m_1, m_2, m_3) acquires the phase $e^{i(m_1+m_2+m_3)\phi}$. The rotation-reduced branch therefore retains only configurations satisfying

$$m_1 + m_2 + m_3 = 0, \qquad m_j \in \{-M, \dots, M\}. \tag{22}$$

The remaining three-radius coefficient is approximated by a low-rank, CP-like factorization,

$$C_{m_1,m_2,m_3}(r_1, r_2, r_3) \approx \sum_{\alpha=1}^{R_3} f_{m_1,\alpha}^{(1)}(r_1) f_{m_2,\alpha}^{(2)}(r_2) f_{m_3,\alpha}^{(3)}(r_3). \tag{23}$$

For a retained configuration $\mathbf{m} = (m_1, m_2, m_3)$, define each projected factor as

$$U_{m,\alpha}^{(q)} = \sum_i f_{m,\alpha}^{(q)}(r_i) e^{im\theta_i} z_i, \qquad q \in \{1,2,3\}. \tag{24}$$

For rank component α, the three factors form a pointwise cubic product,

$$T_{\mathbf{m},\alpha} = U_{m_1,\alpha}^{(1)} U_{m_2,\alpha}^{(2)} U_{m_3,\alpha}^{(3)}. \tag{25}$$

The cubic branch maps the retained products to the output features,

$$P_3 = \mathcal{R}_3\left(\{T_{\mathbf{m},\alpha}\}_{\mathbf{m}\in\mathcal{C}_{3,\alpha}}\right), \tag{26}$$

where $\mathcal{R}_3$ is a learned real-valued projection of the concatenated real and imaginary products. In the default model, $M = 6$ and the canonical nondecreasing zero-sum set contains 25 cubic configurations. The quadratic and cubic ranks are $R_2 = R_3 = 24$. The canonical set avoids storing equivalent permutations as separate configurations; the implementation groups factor projections by absolute harmonic order and accounts for the corresponding factor occurrences.

The reductions are applied sequentially: relative geometry removes redundant absolute coordinates, angular harmonics expose the common-rotation phase law, and low-rank factorization compresses the remaining radial and latent-channel coupling. The fourth-order extension follows the same principle, with the zero-sum condition$m_1 + m_2 + m_3 + m_4 = 0$; P3 remains the default model for the principal comparisons. The complete derivation, including harmonic selection rules, low-rank coefficient forms and the fourth-order extension, is provided in Supplementary Note 4.

From the derivation to the SRPO-P3 network.

The mathematical derivation above describes a scalar local mapping. A practical neural operator must propagate a multi-channel field and learn which latent variables participate in the structured interactions. Given an input tensor $X \in \mathbb{R}^{B\times C_{\text{in}}\times H\times W}$, a 1×1 lifting convolution maps the task-dependent input channels to a hidden field $h^0 \in \mathbb{R}^{B\times 64\times H\times W}$. Five SRPO-P3 blocks are then applied, keeping the persistent hidden width fixed at 64.

Within block ℓ, the hidden field is first normalized with channel-wise RMS normalization and projected pointwise to $K = 20$ learned latent scalar fields, $z^\ell = W_z^\ell \mathrm{RMSNorm}(h^\ell)$. These latent fields are not prescribed physical modes or Fourier modes; they are learned channel-space coordinates that provide a compact basis for the spatial interaction algebra. Each latent field is evaluated through two radial heads, yielding $E = K \times 2 = 40$ effective moment channels.

The polar-harmonic analysis uses a 9×9 base stencil divided into four square-shell radial bins. For each shell, scale, latent mode, radial head and angular order, SRPO learns a radial coefficient multiplying a fixed cosine or sine basis evaluated at the actual grid-offset angle. The same basis is evaluated at five dilation scales, $\mathcal{D} = \{1,2,4,8,16\}$, and scale responses are summed to form a shared harmonic representation. A learnable centre coefficient is added to the $m = 0$ component. The angular order is truncated at $M = 6$, giving one real component for $m = 0$ and cosine/sine components for $m = 1,\ldots,6$, or 13 real angular components per effective moment channel.

For clarity, a discrete cosine kernel for latent mode k, radial head q, harmonic order m and dilation d can be written as

$$K^{c}_{k,q,m,d}(\boldsymbol{\delta}) = \sum_{s=1}^{4} a^{c}_{k,q,m,d,s}\ \mathbf{1}\{\rho(\boldsymbol{\delta}) = s\}\cos(m\theta_{\boldsymbol{\delta}}), \tag{27}$$

where $\boldsymbol{\delta} = (\delta_x, \delta_y)$, $\rho(\boldsymbol{\delta}) = \max(|\delta_x|, |\delta_y|)$ indexes the square shell, and $\theta_{\boldsymbol{\delta}} = \operatorname{atan2}(\delta_y, \delta_x)$. The sine kernel is defined analogously; its $m = 0$ component is zero. This discretization follows the polar construction while aligning with the Cartesian lattice and avoiding interpolation.

The harmonic tensor feeds four parallel branches: the first-order polar branch P_1, the low-rank quadratic branch P_2, the low-rank cubic branch P_3, and a pointwise branch that mixes channels without spatial aggregation. The first-order branch concatenates Z_0 with the real and imaginary components of $Z_1, \ldots, Z_6$, then projects the resulting $13 \times 40 = 520$ channels to 64. Learnable gates on non-zero angular orders are initialized at zero, allowing directional contributions to enter gradually during training. Before the explicit products in P_2 and P_3, the complex moments are passed through a phase-preserving amplitude squash,

$$\tilde{Z}_m = \frac{Z_m}{\sqrt{1 + |Z_m|^2/c^2}}, \qquad c = 3. \tag{28}$$

The denominator is real and shared by the real and imaginary parts, so the phase is preserved while the magnitude is smoothly bounded. The magnitude is computed in FP32 for numerical stability. In the quadratic branch, two rank-24 complex projections are formed for each of the seven stored harmonic orders, multiplied as $U_m\overline{V_m}$, concatenated as real/imaginary channels (336 channels in total), and projected to 64 channels. In the cubic branch, the 25 zero-sum configurations and rank 24 produce 1,200 real/imaginary feature channels before projection to 64. These are the explicit interaction features; the pointwise branch supplies complementary local channel mixing.

The four branch outputs are summed to form the operator update O^{ℓ}. A channel-wise learnable residual scale Γ_{ℓ}, initialized to 0.08, controls the operator step. A second residual stage applies RMSNorm followed by a SwiGLU channel MLP with expansion factor 2; its learnable scale Λ_{ℓ} is also initialized to 0.08:

$$\bar{h}^{\ell} = h^{\ell} + \Gamma_{\ell} \odot O^{\ell}, \qquad h^{\ell+1} = \bar{h}^{\ell} + \Lambda_{\ell} \odot W_{\text{out}}\left[\text{SiLU}\left(A^{\ell}\right) \odot B^{\ell}\right], \tag{29}$$

where $[A^{\ell}, B^{\ell}]$ is the channel-wise split of the $64 \to 256$ projection of $\text{RMSNorm}(\bar{h}^{\ell})$, and W_{out} maps the 128-channel gated features back to 64 channels. After five blocks, the output head applies RMSNorm, a 1×1 projection from 64 to 64 channels, SiLU and a final 1×1 projection to the task-specific output channels. The Navier–Stokes protocol predicts a residual increment that is added to the most recent input frame; the other tasks use the target convention specified by their benchmark loader.

Efficient implementation and boundary handling.

The implementation preserves the mathematical operator while reorganizing its computation for GPU execution. Complex quantities are stored as real/imaginary pairs. Radial-harmonic kernels are assembled by tensor contractions, and the five dilation scales are evaluated with grouped dilated convolutions. The 7 quadratic harmonic projections are fused into a grouped projection; the 75 conceptual cubic factor projections (25 configurations times 3 factors) are grouped by absolute harmonic order into seven larger operations. The resulting quadratic and cubic feature tensors are then formed by pointwise products and projections. Algebraically equivalent convolutional and matrix-multiplication implementations use the same learnable parameters. Circular padding is used for the periodic Navier–Stokes benchmark; non-periodic tasks use the boundary extension associated with their benchmark protocol. The default SRPO-P3 model has approximately 1.70 million trainable parameters, with small task-dependent differences due to input and output projections. Additional implementation details and tensor-shape information are provided in Supplementary Note 1 and Supplementary Table 1.1×1

Benchmarks and forecasting tasks.

We evaluate five two-dimensional PDE benchmarks spanning reaction–diffusion, chaotic incompressible flow, nonlinear free-surface propagation and heterogeneous wave propagation. All are represented on grids. For DiffReact and SWE, we use the PDEBench datasets;[32] Navier–Stokes at uses the released 2D_NS_Re5000 trajectories;[50] Wave-Gauss and Wave-Layer use the PDEGym/Poseidon benchmark family.[51] Dataset provenance, split definitions, input/target construction and sample counts are detailed in Supplementary Note 2 and Supplementary Table 2.$128 \times 128 Re = 5000$

For DiffReact, the 1,000 trajectories are split by trajectory into 800/100/100 training/validation/test trajectories. Each input contains 16 historical two-channel states plus two coordinate channels (34 input channels), and the target is the next two-channel state. Training and validation windows use stride 4, producing 17,600 and 2,200 samples; test windows use stride 1, producing 8,500 samples.

For Navier–Stokes, 100 trajectories at $Re = 5000$ are split 80/10/10 at the trajectory level using a fixed seed. The model receives four consecutive vorticity snapshots and predicts the next snapshot. Training windows use stride 1 (39,760 samples), whereas validation and test windows use stride 2 (2,490 samples each). Normalization statistics are estimated from training trajectories only.

For SWE, the radial dam-break dataset contains 1,000 trajectories, split in the original ordering into 800/100/100. Four consecutive water-depth fields are used to predict the next field. Training and validation windows use stride 4, yielding 20,000 and 2,500 samples; test windows use stride 1, yielding 9,700 samples.

For Wave-Gauss, we use the first 1,024 trajectories from the official training pool and retain the official 60 validation and 240 test trajectories. For Wave-Layer, we use the first 3,072 training trajectories and the same official validation and test sets; only the benchmark's 15-snapshot protocol is used even when raw chunks contain additional frames. Both wave tasks use two previous wave fields, the static propagation-speed field and two coordinate channels as a five-channel input, and predict the next wave field. Six supervised prediction pairs are constructed per trajectory, giving 6,144/360/1,440 training/validation/test samples for Wave-Gauss and 18,432/360/1,440 for Wave-Layer. For all datasets, trajectories are separated before windows are formed, and any normalization statistics are computed from training data only.

Exact trajectory indices, split files and preprocessing details for all five benchmarks are provided in Supplementary Note 2.

Baselines, optimization and evaluation.

The baselines are Full-FNO, ACT on a matched Full-FNO backbone, the official vanilla CNO, and CNO-Large. Full-FNO uses four spectral stages and a full-spectrum representation with a pointwise path; its hidden width is 32 on all five datasets. ACT inserts the authors' released MultiHeadACTBlock after each FNO stage (four heads, hidden dimension 16 and maximum displacement 0.5). CNO uses the official CNO2d_simplified architecture with four layers, four residual blocks, four residual neck blocks, latent dimension 64, no batch normalization and bicubic resampling with antialiasing. CNO-Large changes only the CNO channel multiplier from 16 to 48, serving as a capacity-scaled control.$rFFT21 \times 1$

The main comparison uses seed 2026 and an 80-epoch training horizon. Within each benchmark, all methods share the same trajectory split, input/output construction, validation metric, held-out test set and checkpoint-selection rule. The best checkpoint is selected using validation data before the test set is evaluated. The logical batch size is fixed within each benchmark; gradient accumulation is used when needed to fit the model in memory without changing the effective batch size. Optimization is not forced to be identical across architectures: CNO and CNO-Large use their stable model-specific schedules and strict FP32 where required; SRPO, Full-FNO and ACT use FP16 automatic mixed precision for DiffReact, Navier–Stokes and SWE, while both wave tasks are run in

FP32. The comparison is therefore task- and evaluation-matched, not an optimizer-identical or wall-clock-matched benchmark. The SRPO-P4 runs across all five datasets were executed on an NVIDIA RTX PRO 6000 GPU (90 GB VRAM), while the other model runs used NVIDIA GeForce RTX 5090 GPUs (32 GB VRAM). Full baseline configurations, parameter counts, training/checkpoint protocols, optimizer settings and computing environment are documented in Supplementary Note 3 and Supplementary Tables 3–7.

We retain each benchmark's primary metric: physical relative L_2 error for DiffReact and Navier–Stokes, dense mean-sample NRMSE for SWE, and teacher-forced pair relative L_2 error for Wave-Gauss and Wave-Layer. For a held-out target $\mathbf{y}_n$ and prediction $\hat{\mathbf{y}}_n$, the sample relative L_2 error is

$$\mathrm{RelL2}(\hat{\mathbf{y}}_n, \mathbf{y}_n) = \frac{\| \hat{\mathbf{y}}_n - \mathbf{y}_n \|_2}{\| \mathbf{y}_n \|_2}. \tag{30}$$

Metrics are aggregated according to the formal protocol of each benchmark; lower values indicate better predictions. For the wave tasks, autoregressive rollouts are additionally reported as a distinct evaluation of repeated prediction, while checkpoint selection uses teacher-forced pair error. Because the complete five-model comparison is available for seed 2026 rather than a uniform multi-seed sweep, we report the executed point estimates and do not make mean–standard-deviation or statistical-significance claims. Parameter count is reported as model size and is not treated as a proxy for wall-clock efficiency.

Data Availability

The data supporting the findings of this study are available from the original public benchmark sources. The DiffReact and shallow-water equation (SWE) datasets are available through PDEBench (https://darus.uni-stuttgart.de/dataset.xhtml?persistentId=doi:10.18419/darus-2986). The Navier–Stokes dataset at Reynolds number 5000 is available from Zenodo (https://doi.org/10.5281/zenodo.7495555). The Wave-Gauss and Wave-Layer datasets are available from the official PDEgym/Poseidon dataset repositories (https://huggingface.co/datasets/camlab-ethz/Wave-Gauss and https://huggingface.co/datasets/camlab-ethz/Wave-Layer, respectively). The trajectory splits, preprocessing procedures, input–output construction, and sample counts used in this study are documented in Supplementary Note 2 and Supplementary Table 2. The benchmark datasets are not redistributed in the accompanying code archive and should be obtained from their original sources.

Code Availability

The source code implementing SRPO-P3 and SRPO-P4, together with the training and evaluation scripts, configuration files, and instructions required to reproduce the experiments reported in this study, is provided in the anonymized code archive submitted with the manuscript for peer review. The code will be made publicly available in a public repository upon publication of the article, and the repository link will be provided in the published version. The benchmark datasets are not included in the code archive and can be obtained from their original sources listed in the Data Availability statement.

References

1. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
2. Li, Z. et al. Fourier neural operator for parametric partial differential equations. In Proc. International Conference on Learning Representations (ICLR, 2021).
3. Kovachki, N. et al. Neural operator: Learning maps between function spaces with applications to PDEs. *J. Mach. Learn. Res.* **24**, 1–97 (2023).
4. Raonić, B. et al. Convolutional neural operators for robust and accurate learning of PDEs. *Adv. Neural Inf. Process. Syst.* **36**, 77187–77200 (2023).
5. Rahman, M. A., Ross, Z. E. & Azizzadenesheli, K. U-NO: U-shaped neural operators. *Trans. Mach. Learn. Res.* https://openreview.net/forum?id=j3oQF9coJd (2023).
6. Tran, A., Mathews, A., Xie, L. & Ong, C. S. Factorized Fourier neural operators. In Proc. International Conference on Learning Representations (ICLR, 2023).
7. Hao, Z. et al. GNOT: A general neural operator transformer for operator learning. In Proc. 40th International Conference on Machine Learning, PMLR 202, 12556–12569 (2023).
8. Wu, H., Luo, H., Wang, H., Wang, J. & Long, M. Transolver: A fast transformer solver for PDEs on general geometries. In Proc. 41st International Conference on Machine Learning, PMLR 235, 53681–53705 (2024).
9. Li, Z., Huang, D. Z., Liu, B. & Anandkumar, A. Fourier neural operator with learned deformations for PDEs on general geometries. *J. Mach. Learn. Res.* **24**, 1–26 (2023).
10. Helwig, J. et al. Group equivariant Fourier neural operators for partial differential equations. In Proc. 40th International Conference on Machine Learning, PMLR 202, 12907–12930 (2023).
11. Cao, S. Choose a Transformer: Fourier or Galerkin. *Adv. Neural Inf. Process. Syst.* **34**, 24924–24940 (2021).
12. Cao, Q., Goswami, S. & Karniadakis, G. E. Laplace neural operator for solving differential equations. *Nat. Mach. Intell.* **6**, 631–640 (2024).
13. Zappala, E. et al. Learning integral operators via neural integral equations. *Nat. Mach. Intell.* **6**, 1046–1062 (2024).
14. Zhang, E. et al. Blending neural operators and relaxation methods in PDE numerical solvers. *Nat. Mach. Intell.* **6**, 1303–1313 (2024).

15. Berner, J. et al. Principled approaches for extending neural architectures to function spaces for operator learning. *Nat. Mach. Intell.* **8**, 1173–1181 (2026).

16. Fabiani, G., Vandecasteele, H., Goswami, S., Siettos, C. & Kevrekidis, I. G. Enabling local neural operators to perform equation-free system-level analysis. *Nat. Mach. Intell.* **8**, 1127–1141 (2026).

17. Long, Z., Zhou, Q., Zhu, A., Dai, D. & Liu, Y. Deep neural operator for free boundary problems. *Nat. Mach. Intell.* **8**, 806–817 (2026).

18. Lingsch, L. E. et al. Beyond regular grids: Fourier-based neural operators on arbitrary domains. In Proc. 41st International Conference on Machine Learning, PMLR 235, 30610–30629 (2024).

19. Liu-Schiaffini, M. et al. Neural operators with localized integral and differential kernels. In Proc. 41st International Conference on Machine Learning, PMLR 235, 32576–32594 (2024).

20. Cheng, Z. et al. Reference neural operators: Learning the smooth dependence of solutions of PDEs on geometric deformations. In Proc. 41st International Conference on Machine Learning, PMLR 235, 8060–8076 (2024).

21. Liu, C. et al. Adaptive coordinate transforms for neural operators. Preprint at https://arxiv.org/abs/2605.06203 (2026).

22. Liu, S., Yang, X. & Chen, Y. Riesz neural operator for solving partial differential equations. In Proc. International Conference on Learning Representations (ICLR, 2026).

23. Huang, J., Zhang, K., Wu, Y. & Cheng, Z. Operator learning with domain decomposition for geometry generalization in PDE solving. In Proc. International Conference on Learning Representations (ICLR, 2026).

24. Bonev, B. et al. Spherical Fourier neural operators: Learning stable dynamics on the sphere. In Proc. 40th International Conference on Machine Learning, PMLR 202, 2806–2823 (2023).

25. Hu, P. et al. Wavelet diffusion neural operator. In Proc. International Conference on Learning Representations (ICLR, 2025).

26. Buitrago Ruiz, R., Marwah, T., Gu, A. & Risteski, A. On the benefits of memory for modeling time-dependent PDEs. In Proc. International Conference on Learning Representations (ICLR, 2025).

27. Hou, X., Huang, X. & Perdikaris, P. CFO: Learning continuous-time PDE dynamics via flow-matched neural operators. In Proc. International Conference on Learning Representations (ICLR, 2026).

28. Ma, S. et al. Learning data-efficient and generalizable neural operators via fundamental physics knowledge. In Proc. International Conference on Learning Representations (ICLR, 2026).

29. Zhang, Z., Wu, Y., Zhang, K. & Wang, Y. From cheap geometry to expensive physics: A physics-agnostic pretraining framework for neural operators. In Proc. International Conference on Learning Representations (ICLR, 2026).

30. Chen, W. et al. Data-efficient operator learning via unsupervised pretraining and in-context learning. *Adv. Neural Inf. Process. Syst.* **37**, 6213–6245 (2024).

31. Ohana, R. et al. The Well: A large-scale collection of diverse physics simulations for machine learning. *Adv. Neural Inf. Process. Syst.* **37**, Datasets and Benchmarks Track (2024).

32. Takamoto, M. et al. PDEBench: An extensive benchmark for scientific machine learning. *Adv. Neural Inf. Process. Syst.* **35**, 1596–1611 (2022).

33. Lippe, P., Veeling, B., Perdikaris, P., Turner, R. E. & Brandstetter, J. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. *Adv. Neural Inf. Process. Syst.* **36**, 67398–67433 (2023).

34. Li, K. & Ye, W. D-FNO: A decomposed Fourier neural operator for large-scale parametric partial differential equations. *Comput. Methods Appl. Mech. Eng.* **436**, 117732 (2025).

35. Chen, J. et al. Enforcing the principle of locality for physical simulations with neural operators. *J. Comput. Phys.* **538**, 114131 (2025).

36. Zhou, X.-H. et al. Neural operator-based super-fidelity: A warm-start approach for accelerating steady-state simulations. *J. Comput. Phys.* **529**, 113871 (2025).

37. Song, S., Mukerji, T. & Zhang, D. Physics-informed multi-grid neural operator: Theory and an application to porous flow simulation. *J. Comput. Phys.* **520**, 113438 (2025).

38. Li, H., Ye, X., Jiang, P., Qin, G. & Wang, T. Local neural operator for solving transient partial differential equations on varied domains. *Comput. Methods Appl. Mech. Eng.* **427**, 117062 (2024).

39. Xiong, W. et al. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. *J. Comput. Phys.* **513**, 113194 (2024).

40. Shi, P., Yin, J., Tierney, S. & Gao, J. SirenFNO: Efficient and full frequency learning of Fourier neural operators. In Proc. 35th International Joint Conference on Artificial Intelligence, 4804–4812 (IJCAI, 2026).

41. Gopakumar, V. et al. Learning physical operators using neural operators. In Proc. 29th International Conference on Artificial Intelligence and Statistics, PMLR 300, 3223–3231 (2026).

42. Chrysos, G. G. et al. P-nets: Deep polynomial neural networks. In Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7325–7335 (2020).

43. Batatia, I., Kovács, D. P., Simm, G., Ortner, C. & Csányi, G. MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. *Adv. Neural Inf. Process. Syst.* **35**, 11423–11436 (2022).

44. Sanborn, S., Shewmake, C., Olshausen, B. & Hillar, C. Bispectral neural networks. In Proc. International Conference on Learning Representations (ICLR, 2023).

45. Cohen, T. & Welling, M. Group equivariant convolutional networks. In Proc. 33rd International Conference on Machine Learning, PMLR 48, 2990–2999 (2016).

46. Weiler, M. & Cesa, G. General E(2)-equivariant steerable CNNs. *Adv. Neural Inf. Process. Syst.* **32**, 14334–14345 (2019).

47. Shumaylov, Z. et al. Lie algebra canonicalization: Equivariant neural operators under arbitrary Lie groups. In Proc. International Conference on Learning Representations (ICLR, 2025).

48. Lu, L. et al. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Comput. Methods Appl. Mech. Eng.* **393**, 114778 (2022).

49. Kontolati, K., Goswami, S., Shields, M. D. & Karniadakis, G. E. On the influence of over-parameterization in manifold based surrogates and deep neural operators. *J. Comput. Phys.* **479**, 112008 (2023).

50. Li, Z. et al. Learning chaotic dynamics in dissipative systems. Adv. Neural Inf. Process. Syst. 35, 16768–16781 (2022).

51. Herde, M. et al. Poseidon: Efficient foundation models for PDEs. Adv. Neural Inf. Process. Syst. 37 (2024).

Competing Interests

The authors declare no competing interests.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SRPOSupplementaryInformation.docx