

Reformulating Neural Operators in $d + 1$ Dimensions for Embedding Evolution

Haoze Song¹ Zhihao Li¹ Xiaobo Zhang³ Zecheng Gan^{1,2} Zhilu Lai^{1,2} Wei Wang^{1,2}
¹HKUST (GZ) ²HKUST ³SWJTU
 {hsong492, zli416}@connect.hkust-gz.edu.cn
 zhangxb@swjtu.edu.cn {zechenggan, zhilulai, weiwcs}@ust.hk

Abstract

Neural Operators (NOs) are powerful architectures for learning mappings between function spaces. While most advances focus on refining kernel parameterizations over the d -dimensional physical domain, the evolution of lifted embeddings remains underexplored, which often drives models toward computationally expensive embedding-scaling designs to improve approximation.

In this paper, we introduce an auxiliary function dimension that models embedding evolution in operator form, thereby reformulating the NO pipeline in $d + 1$ dimensions. We instantiate this framework via Fourier-based operators acting jointly on the physical and auxiliary domains, yielding a basis-diversified auxiliary evolution module as an alternative to brute-force embedding scaling. Across more than ten increasingly challenging benchmarks, ranging from the 1D heat equation to the highly nonlinear 3D Rayleigh–Taylor instability, our model consistently achieves the lowest relative L_2 error among the evaluated baselines. Crucially, this advantage is empirically supported by (1) controlled budget-aware comparisons against scaled and ablated baselines; (2) robustness under mixed-resolution training and super-resolution inference; and (3) zero-shot generalization to unseen temporal regimes. In addition, we present a broader set of design choices for lifting and recovery operators, demonstrating their impact on our model’s predictive performance.

1 Introduction

Neural Operators (NOs) [34, 36, 26, 29, 64, 37, 50, 66, 39, 32] parameterize mappings between function spaces. As a new paradigm for learning solution operators of Partial Differential Equations (PDEs), they often improve accuracy relative to standard neural networks [34, 32].

A natural function-space analogue of matrix multiplication in standard Neural Networks [17, 30] is kernel integration. Concretely, NOs define kernel integrals acting on embedding functions defined on the d -dimensional physical domain, whose pointwise values are lifted from the input at each location [34]. Building on this formulation, successive variants have improved the parameterization of these kernel integrals through richer aggregation mechanisms, such as graph aggregation [35, 32], spectral convolution [36, 62], and attention [65, 67].

However, when the lifted embedding is not sufficiently expressive, existing architectures [33, 65, 32] typically compensate by enlarging the embedding width or adding heads, while leaving the mechanism by which embeddings evolve largely unchanged. This strategy becomes computationally expensive as the embedding size grows, since dense channel mixing scales quadratically with the embedding width, while head-wise factorization only partially alleviates this cost, since it induces a block-diagonal interaction structure that weakens direct cross-head coupling [60, 6].

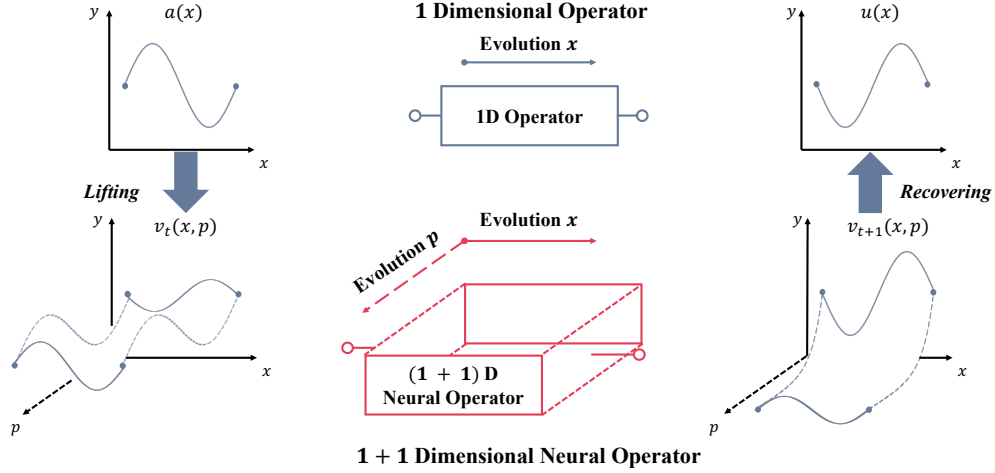


Figure 1: Different d -dimensional operators can be parameterized by the $(d + 1)$ -dimensional Neural Operator which flexibly represents the evolution on both x and p .

As a result, improving expressivity often relies on brute-force embedding enlargement rather than a more direct design of how embeddings evolve. This raises a natural question:

Can we improve how embeddings evolve by applying the operator-design principle along an auxiliary embedding coordinate, instead of merely increasing the embedding width?

In this paper, we answer this question by introducing an auxiliary embedding dimension p , and reformulating the latent NO pipeline on the $(d + 1)$ -dimensional product domain: instead of only updating embeddings over the physical domain, the proposed formulation evolves latent functions over both the physical and the auxiliary embedding coordinate.

This auxiliary-domain propagation provides an explicit operator-based mechanism for embedding evolution. In our Fourier-based instantiation, the auxiliary profile is updated through two coordinate views: raw p -coordinate mixing and Fourier- p -coordinate mixing. We refer to this as basis-diversified auxiliary evolution, and evaluate it through controlled scaled and ablated baselines. Specifically, our main contributions are summarized as follows:

- We reformulate the latent NO pipeline to evolve via kernel integral operators over both physical and auxiliary coordinates, establishing an operator-level mechanism for embedding evolution. We instantiate this framework with Fourier-based kernels into a specific model, the Schrödingerised Kernel Neural Operator (SKNO), whose auxiliary update mixes the lifted embedding in both raw- p and Fourier- p coordinates.
- We evaluate SKNO across more than ten benchmarks spanning linear, nonlinear, high-dimensional, and spatio-temporal dynamics. Across these tasks, SKNO consistently achieves the lowest relative L_2 error among the evaluated baselines. Controlled comparisons accounting for parameters, FLOPs, and wall-clock time, together with ablated and scaled ablations, indicate that the gain is not explained by additional same-basis capacity alone. Further tests on zero-shot super-resolution, temporal generalization, lifting, recovery, and auxiliary-domain operators support the robustness and scalability of the proposed framework.

2 Neural Operator in $d + 1$ Dimensions

Problem setup. Let $D_x \subset \mathbb{R}^d$ be a bounded physical domain. Given an input field $a(x) \in \mathcal{A}(D_x; \mathbb{R}^{d_a})$, our goal is to learn an operator

$$\mathcal{G} : a \mapsto u = \mathcal{G}[a], \quad (1)$$

where $u(x) \in \mathcal{U}(D_x; \mathbb{R}^{d_u})$. Here $x = (x_1, \dots, x_d)$ denotes the physical coordinate, and $\mathcal{A}$ and $\mathcal{U}$ are (suitable subsets of) Banach spaces, where functions a and u take values in $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_u}$ ($d_a, d_u \in \mathbb{N}$), respectively.

Auxiliary function dimension. Previous neural operators first lift the input to a vector-valued embedding field, and then evolve it over the physical domain D_x . In our framework, we take a different view. We introduce an auxiliary domain $D_p \subset \mathbb{R}$ and lift the input to a scalar latent function $v_0(x, p) \in \mathcal{V}(D_x \times D_p; \mathbb{R})$. The variable p is an auxiliary embedding coordinate. Figure 8 summarizes this design distinction between previous NO variants and ours.

Lifting. To prepare the $(d+1)$ -dimensional scalar function $v(x, p) \in \mathcal{V}(D_x \times D_p; \mathbb{R})$ for subsequent evolving, the lifting operator $\mathcal{P}$ first maps the input field $a(x)$ to a latent function on the product domain $D_x \times D_p$:

$$v_0(x, p) := \mathcal{P}[a](x, p). \quad (2)$$

A simple separated lifting is

$$v_0(x, p) = w^\top(p)a(x), \quad (3)$$

where $w(p) \in \mathbb{R}^{d_a}$ is usually learnable. If positional features are used, they are concatenated with $a(x)$ before lifting.

$(d+1)$ -dimensional evolution. The lifted function is evolved by iterating learnable linear operators $\mathcal{L}$ and nonlinear maps σ , where the core block is the kernel integral operator $\mathcal{K}$. For the l -th updated latent function $v_l(x, p)$, we define the kernel integral operator on $D_x \times D_p$ as follows:

$$\mathcal{K}_l[v_l](x, p) := \int_{D_x} \int_{D_p} \kappa_l(x, y, p, p') v_l(y, p') dp' dy, \quad (4)$$

where $\kappa_l : \mathbb{R}^{2(d+1)} \rightarrow \mathbb{R}$ is the l -th learnable kernel function and $l = 0, 1, \dots, L-1$ for κ .

Recovery. After the latent function has evolved, the recovery operator $\mathcal{Q}$ maps it back to the original output domain through:

$$u_{\text{pred}}(x) := \mathcal{Q}[v_L](x) = \int_{D_p} \chi(p) v_L(x, p) dp, \quad (5)$$

where $\chi(p) \in \mathbb{R}^{d_u}$ is usually learnable.

Given all the definitions introduced above, we assemble our neural operator framework to approximate $\mathcal{G}$ in the problem setup as follows:

$$\begin{aligned} \mathcal{G}[a] &\approx (\mathcal{Q} \circ \mathcal{M} \circ \mathcal{P})[a] \\ &= (\mathcal{Q} \circ (\sigma_{L-1} \circ \mathcal{L}_{L-1}) \circ \dots \circ (\sigma_0 \circ \mathcal{L}_0) \circ \mathcal{P})[a], \end{aligned} \quad (6)$$

Here, each $\mathcal{L}_l$ denotes a general learnable $(d+1)$ -dimensional linear operator, and σ_l can be a nonlinear activation function or a shallow MLP, providing the nonlinear composition needed to approximate nonlinear target operators $\mathcal{G}$.

3 Schrödingerised Kernel Neural Operator

We now instantiate the $(d+1)$ -dimensional framework in Section 2 with a Fourier-based model, called the Schrödingerised Kernel Neural Operator (SKNO). Specifically, SKNO evolves the lifted latent function $v(x, p)$ by using Fourier-based operators over the physical coordinate x and the auxiliary p .

Naming. The term ‘‘Schrödingerised’’ reflects the use of Fourier-domain propagation on the introduced auxiliary phase-like coordinate as in the Schrödingerised solver. In our work, Schrödingerisation ONLY serves as design inspiration for building structured operator evolution along the auxiliary coordinate.

An overview of the SKNO architecture is shown in Figure 2. Beyond the lifting and recovering modules, SKNO is built from signal propagators that instantiate the kernel integration in Eq. (6) through structured physical-domain and auxiliary-domain updates, together with bias terms and residual connections.

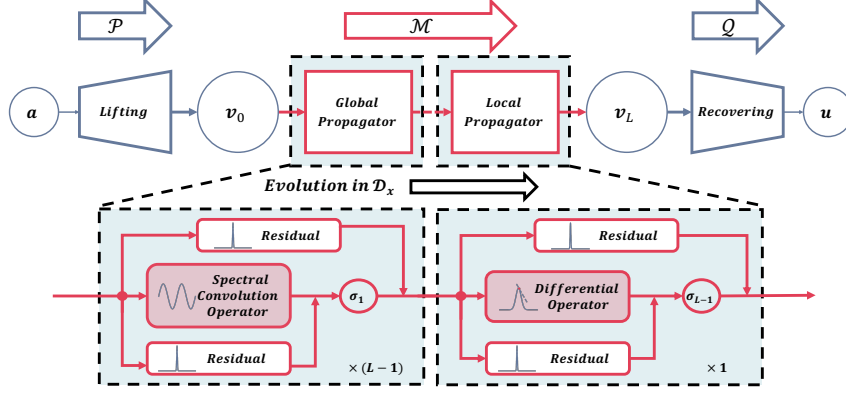


Figure 2: Overview of the Schrödingerised Kernel Neural Operator (SKNO) architecture. The input signal $\mathbf{a}$ is lifted to a $(d + 1)$ -dimensional signal $\mathbf{v}_0$ through the preparation lifting module. Such a $(d + 1)$ -dimensional signal evolves through $(L - 1)$ global propagators and one local propagator in D_x , capturing both global and local signal changes. The measuring module recovers the output signal $\mathbf{u}$ from evolved $\mathbf{v}_L$.

Lifting and Recovery. We consider several choices for $\mathcal{P}$ and $\mathcal{Q}$, and evaluate their effects in Section 4.2. Unless otherwise stated, SKNO uses a pointwise linear layer for the separated lifting map in Eq. (1), which balances representation flexibility and computational efficiency. For recovery, we use an MLP-based projection whose final linear layer implements the quadrature over D_p . This choice keeps the lifting/recovering modules comparable to common NO implementations, so that the main architectural difference lies in the proposed $(d + 1)$ -dimensional evolution.

Signal propagation on D_x . As illustrated in Figure 2, we employ $(L - 1)$ global propagators and one last local propagator for modeling the underlying $(d + 1)$ -dimensional function evolution on D_x .

Spectral Convolution Operator. Differential operators with respect to x , such as ∂_x and ∂_{xx} , can be diagonalized in Fourier domain of x according to the properties of Fourier Transform (FT). Based on that, we leverage the implementation of spectral convolution operator in [36], which uses a learnable complex vector equivalent to a diagonal matrix in the Fourier domain of D_x . In practice, the complex vector is truncated for working on different grid sizes:

$$\text{SpectralConvOp}(\mathbf{v}_l) = F_x^{-1}(\text{trun_diag}(F_x(\mathbf{v}_l))), \quad (7)$$

where $\mathbf{v}_l$ denotes the input latent signal fitting $v(x, p)$, and F_x and F_x^{-1} denote Discrete Fourier Transform (DFT) and Inverse DFT (accelerated by Fast Fourier Transform (FFT) in practice). The `trun_diag()` represents the element-wise multiplication between a learnable truncated complex vector and signal slices of $\mathbf{v}_l$ on the transformed grid ξ after F_x .

Differential Operator. Since the truncation in *Spectral Convolution Operator* prefers to propagate signals in a relatively global scope, we compensate for possible local information lost by propagating the finite difference derivative of x on the local region based on [39]:

$$\mathcal{K}_{L-1}[\mathbf{v}_{L-1}](x, p) = \frac{\iint_{D_{loc} \times D_p} \kappa_{L-1}(x, y, p, p') (v_{L-1}(y, p') - v_{L-1}(x, p')) dy dp'}{\iint_{D_{loc} \times D_p} \kappa_{L-1}(x, y, p, p') (y - x, p') dy dp'}, \quad (8)$$

where $D_{loc} \subset D_x$ denotes the local region around the query position x . Related ablation results are reported in Tab. 3 and Tab. 19.

Evolution along the auxiliary dimension For each spatial location $x_i = (x_{i_1}, \dots, x_{i_d})$, the discretized signal along auxiliary dimension $\mathbf{v}_l(x_i, \cdot) \in \mathbb{R}^{N_p}$ is updated by a basis-diversified auxiliary operator:

$$\mathbf{v}_l(x_i, \cdot) \mapsto (F_p^{-1} \tilde{A}_l F_p + B_l) \mathbf{v}_l(x_i, \cdot),$$

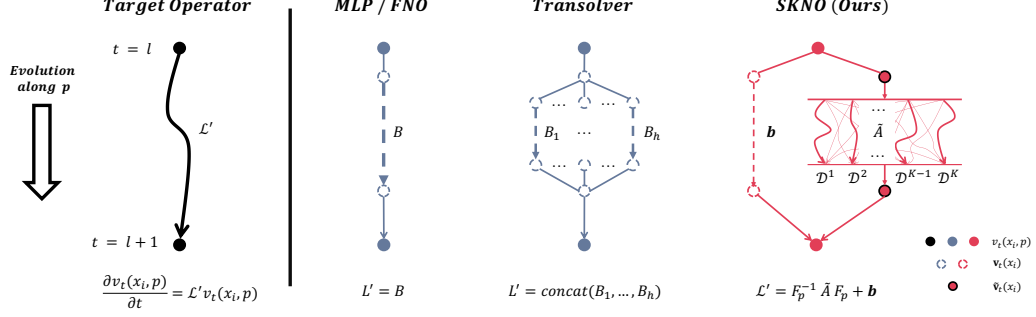


Figure 3: Evolution along the auxiliary dimension and corresponding neural operator implementations. *Left:* The target operator $\mathcal{L}'$ governs the evolution along p . *Right:* Whereas prior methods rely on discretized matrices at each p location, our model captures this evolution using operator modules acting on dual spaces.

where F_p is the DFT matrix along the auxiliary axis, $\tilde{A}_l$ mixes the signal in Fourier- p coordinates, and B_l mixes it in the raw p -coordinates. This design introduces a finite-budget architectural bias: the two branches parameterize the auxiliary evolution through different coordinate views, and subsequent nonlinear/residual updates can exploit features formed from both views.

Linear block with residuals. With above two implementations of $(d+1)$ -dimensional kernel integral operators, we specified the linear block $\mathcal{L}$ in Eq.6 with two parts, kernel integral operator $\mathcal{K}_l$ and residual operator $\bar{\mathcal{W}}_l$:

$$\mathcal{L}_l[v_l](x, p) = (\mathcal{K}_l + \bar{\mathcal{W}}_l)[v_l](x, p), \quad (9)$$

where $\bar{\mathcal{W}}_l[v_l](x, p) = (\mathcal{I} + \mathcal{W}_l)[v_l](x, p) = v_l(x, p) + \int_{D_p} w_l(p, p') v_l(x, p') dp'$. Here, $\mathcal{W}_l$ is implemented by a shallow MLP and $\mathcal{I}$ is implemented by a residual connection. If moving the residual connection from right to left, Eq. 9 implements a forward finite-difference (Euler) update.

As illustrated in Figure 3, while standard channel-mixing designs update the embedding through a single coordinate view, SKNO updates the auxiliary profile through both raw- p and Fourier- p views. By iteratively combining physical-domain propagation with this auxiliary-domain update, SKNO instantiates the $(d+1)$ -dimensional operator framework while keeping the additional cost controlled through FFT-based operations.

4 Experiments

4.1 General Setting

Benchmarks We evaluate the proposed model on more than ten diverse benchmarks spanning a wide spectrum of PDE dynamics. To highlight the effectiveness of our evolving kernel integral design, we first compare different kernel parameterizations on the PDE operators including the 1D heat equation [55] and the 1D advection equation [56]. We then evaluate full models on a broader set of PDE systems, conducting extensive experiments on PDEs including the 1D Burgers equation; 2D Darcy flow with discontinuous coefficients; 2D incompressible Navier–Stokes at low viscosity ($1e-5$) [36, 65]; 2D Gray–Scott Reaction–Diffusion [23]; and the highly unstable 3D Rayleigh–Taylor Instability [46]. A summary of all benchmarks is in Table 20. Per-benchmark details and additional results are provided in Appendix D.

Baselines and Configurations We compare our model with various d -dimensional neural implementations, including DeepONet [41], Fourier Neural Operator (FNO) [36], Convolutional Neural Operator (CNO) [51], and transformer-based model Transolver [65]. All models are trained using the recommended or reported configurations from their papers and official implementations when available. For benchmarks such as 1D Burgers, 2D Darcy Flow, and 2D Gray–Scott, we train each model for 500 epochs with no physical prior regularization, using only relative L_2 loss. Because the compared architectures differ in parameter count, FLOPs, and training speed, we do not rely

Table 1: Main experimental results across 1D, 2D, and 3D benchmarks. Each benchmark block reports Rel. L_2 Error and Training Time. For 2D Stress and 2D Strain, the training time is the reported average over the two benchmarks.

Model	1D Burgers [36]		2D Stress [53]		2D Strain [53]		2D Darcy [36]	
	Rel. L_2	Time	Rel. L_2	Time	Rel. L_2	Time	Rel. L_2	Time
DeepONet	8.991e−2	2.07 min	3.891e−1	4.15 min	5.439e−1	4.15 min	6.097e−2	5.20 min
FNO	6.479e−4	6.86 min	6.221e−2	9.43 min	5.753e−2	9.43 min	6.201e−3	10.21 min
Transolver	6.277e−3	94.23 min	4.812e−2	117.39 min	4.657e−2	117.39 min	5.853e−3	115.48 min
SKNO	5.475e−4	15.51 min	4.724e−2	19.36 min	4.635e−2	19.36 min	5.555e−3	17.59 min

Model	2D Gray-Scott [23]		2D Inc. NS [36]		3D Comp. NS [56]		3D RT [46]	
	Rel. L_2	Time	Rel. L_2	Time	Rel. L_2	Time	Rel. L_2	Time
DeepONet	1.528e−1	3.01 min	3.448e−1	865.75 min	5.628e−1	3.90 min	5.731e−2	1.56 min
FNO	2.425e−2	9.64 min	1.280e−1	773.33 min	2.631e−1	10.12 min	5.219e−2	5.12 min
Transolver	3.573e−2	234.96 min	1.002e−1	1263.33 min	2.947e−1	361.41 min	5.723e−2	135.03 min
SKNO	1.298e−2	20.11 min	8.717e−2	708.93 min	2.376e−1	23.33 min	4.471e−2	10.31 min

solely on epoch-matched comparisons. For the incompressible 2D Navier–Stokes, we ensure fairness by matching comparable training time scales, accounting for differences in training speed between transformer-based and other operator architectures. All experiments are run on a single *NVIDIA GeForce RTX 4090* GPU. Complete configurations are provided in Table 6.

4.2 Main Results and Analysis

Results SKNO attains the lowest relative L_2 error among the compared baselines across the evaluated benchmarks. Table 2 summarizes the performance of different $\mathcal{L}$ parameterizations across representative models: the Fourier integral operator in FNO, the physics-attention block in Transolver, and SKNO’s spectral convolution operator acting on $(d + 1)$ -dimensional functions. All modules are evaluated with their original lifting and recovering components and without additional positional encoding (see Appendix D).

For the heat equation experiment, the solution at $t = 1$ is predicted with the same truncation settings as in practice (i.e., truncated modes or slices, both of which parameterize aggregation on the retained coefficient or slice subspace). For the advection equation, these kernels are tested in an autoregressive setting to validate performance on sequential prediction with concatenated input quantities; the reported error corresponds to predicting the next 10 time steps autoregressively given the previous 10.

Relative to FNO, SKNO yields an approximately 37.1% relative L_2 error reduction on 2D incompressible Navier–Stokes ($\nu = 10^{-5}$) using the six-seed mean, and a 42.1% reduction on 2D Gray–Scott relative to the default FNO configuration. On 3D Rayleigh–Taylor instability, SKNO attains an approximately 14.3% error reduction, while the compared baselines remain in a similar error range. Across the benchmarks in Table 1, Table 17 and Table 18, SKNO attains the lowest reported error. Visualizations in Figure 13 and Figure 14 further illustrate that SKNO better captures sharp local structures, such as discontinuities at material interfaces and vorticity patterns in low-viscosity fluid flows. For complete experimental settings, case studies, and full results, see Appendix D.

In addition to the main benchmark results, we report parameter counts, FLOPs, wall-clock training time, and controlled scaled/ablated variants where applicable.

Capacity efficiency. To test whether the improvement comes merely from additional embedding capacity, we compare SKNO against systematically scaled FNO variants, parallel stacked FNO variants, and a same-basis SKNO($B + B$) control. The $B + B$ variant keeps the dual-branch structure but replaces the Fourier- p branch with a duplicate raw- p branch. Thus, $B + B$ controls for extra same-basis capacity, while $A + B$ tests the basis-diversified auxiliary update. As shown in Table 10, SKNO($A + B$) achieves the lowest error with fewer parameters and FLOPs than the best scaled FNO configuration, while SKNO($B + B$) fails to match $A + B$ on 2D Gray–Scott. This supports the view that the gain is not explained by widening or duplicating the same channel-mixing path alone.

Table 2: Different $\mathcal{L}$ parameterizations’ performance on learning operators governed by 1D Heat and Advection Equations.

Linear Blocks	Error
1D Heat Equation [55]	
FNO (1 mode, 4 grid p)	9.305e-1
Transolver (1 slice, 4 grid p / 2 heads)	9.060e-1
Transolver (4 slices, 4 grid p / 2 heads)	8.988e-1
Transolver (4 slices, 4 grid p / 1 head)	8.987e-1
Ours (1 mode, 4 grid p)	2.637e-3
1D Advection Equation [56]	
Transolver (8 slices, 16 grid p / 4 heads)	7.118e-1
Transolver (8 slices, 16 grid p / 2 heads)	6.984e-1
Transolver (8 slices, 16 grid p / 1 head)	6.828e-1
FNO (8 modes, 16 grid p)	6.012e-2
Ours (8 modes, 16 grid p)	1.979e-2

Table 3: Ablation study for SKNO on 2D Darcy. The Linear Residual is $\tilde{\mathcal{W}}_l$ in Linear Block $\mathcal{L}$, while the non-linear one is out of $\mathcal{L}$.

Configuration	Error
w.o. Double Res.	1.363e-2
w.o. Linear Res.	8.284e-3
w.o. Non-linear Res.	7.621e-3
w.o. $\tilde{A}$ along p dim	8.804e-3
w.o. b along p dim	6.827e-3
w.o. Global Propagators	1.584e-1
w.o. Local Propagator	5.715e-3
Baseline (SKNO)	5.555e-3

Table 4: The error on 1D Heat and 2D Darcy with different $\mathcal{Q}$ implementations.

$\mathcal{Q}$	Delta	Step	Mean	Linear	MLP	MLP (Dropout)
1D Heat Equation [55]						
Error	2.538e-3	2.538e-3	2.538e-3	2.538e-3	<u>2.637e-3</u>	7.928e-3
2D Darcy Flow [36]						
Error	5.752e-3	1.402e-2	5.717e-3	5.741e-3	5.555e-3	<u>5.678e-3</u>

Resolution Invariance A key advantage of neural operators is their ability to generalize across discretizations. To evaluate this property, we design experiments where training data of mixed resolutions are combined. Table 17 shows that SKNO consistently achieves the best accuracy, even under randomly mixed training resolutions, demonstrating robustness. In addition, we further evaluate super-resolution performance on the 1D Burgers equation, 2D Darcy flow, and the challenging ERA5 wind field prediction task. All models are trained on a fixed resolution and tested on lower and higher ones. As shown in Tables 15 and 16, SKNO consistently achieves lower relative L_2 errors across resolutions than competing models, whereas the others exhibit substantial error fluctuations as resolution increases. This indicates their instability in generalizing across resolutions. In contrast, SKNO delivers more stable and reliable predictions across varying scales.

Different Lifting and Recovery Operators We explore a broader set of lifting and recovery operators to test their impact on the output solution. Specifically, we fix $\mathcal{P}$ as a linear layer and vary the integration weights $\chi(p)$ for $\mathcal{Q}$. Table 4 shows that all recovering designs perform well on the heat equation, where the Mean and Linear can also be properly scaled by the symmetry of $w(p)$. Some predefined recovery operators also remain effective on the more complex Darcy cases, suggesting that the learned $(d+1)$ -dimensional evolution can provide useful latent structure before the final integration over D_p . Combining results from Table 4 and Table 5 on 2D Darcy, we find that for functions with complex topology on D_x , a nonlinear transformation on $(d+1)$ -dimensional signals, either after the first lifting or before the last recovery, helps adjust the propagated $(d+1)$ -dimensional latent structure before recovery. The best performance with the MLP–Linear combination is consistent with the MLP–Linear implementation in Transolver [65] for handling complex topology.

Ablation We conduct a detailed ablation study on 2D Darcy (Table 3) to evaluate the contribution of different components. Removing both linear and nonlinear residual layers causes a significant performance degradation, indicating their importance for modeling complex $(d+1)$ -dimensional evolution. Omitting either the Fourier- p branch $\tilde{A}$ or the raw- p branch B also increases the error, suggesting that both coordinate views are useful in this setting. Finally, we separate the contributions

of the global and local propagators: the global propagator captures long-range dependencies, while the local propagator refines predictions in regions with sharp transitions, as reflected by the increase in error when the corresponding component is removed.

Table 5: The error on 2D Darcy with different $\mathcal{P}$, $\mathcal{Q}$ combinations.

$\mathcal{Q} \backslash \mathcal{P}$	Constant	Linear	MLP	MLP (Dropout)
Linear	8.738e-3	5.741e-3	5.952e-3	5.451e-3
MLP	8.791e-3	5.555e-3	6.614e-3	5.845e-3

Other Experiments Additional experimental results and visualizations are provided in the Appendix, mainly including (i) zero-shot generalization to unseen temporal regimes: SKNO maintains the lowest error among the compared baselines on 2D incompressible Navier–Stokes in Figure 4; and (ii) other operators choices along p : physics-attention [65] also benefits from adopting the $(d + 1)$ -dimensional operator design, see Table 13.

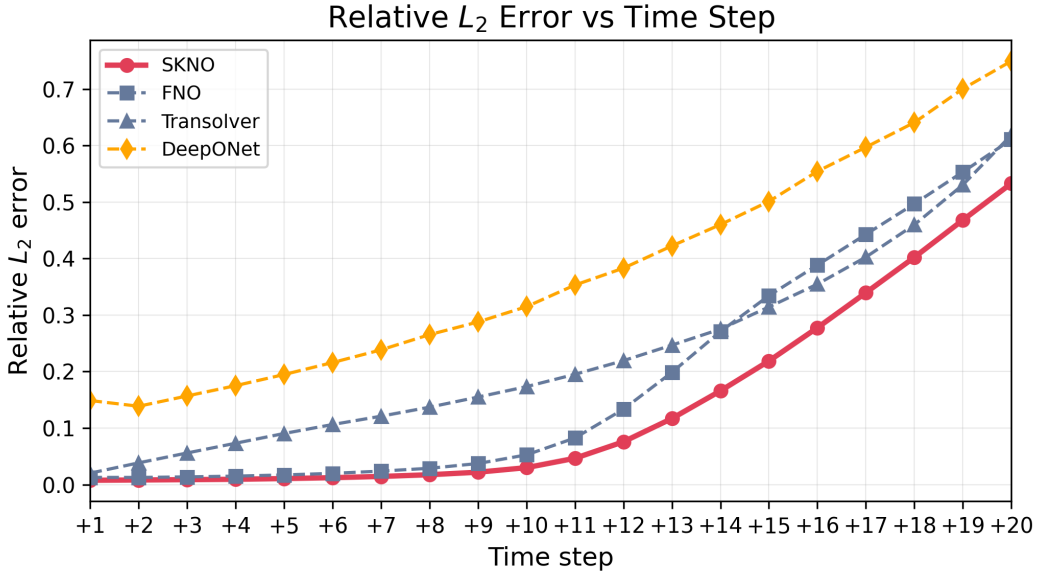


Figure 4: Visualization of zero-shot predictions at future time steps. For the 2D incompressible Navier–Stokes with viscosity 1×10^{-4} [36], all models are autoregressively evaluated over the twenty time steps following the input time. SKNO maintains its better performance on both the seen temporal range (+1 to +10) and the unseen temporal range (+11 to +20), achieving a substantial performance margin over competing methods. Compared with FNO, Transolver, and DeepONet, SKNO achieves average error reductions of 34.5%, 58.7%, and 74.5%, respectively, across all shown predictions. Note that even on the first predicted step, SKNO attains an error of 0.0076, outperforming FNO (0.0127), Transolver (0.0202), and DeepONet (0.1488).

5 Conclusion, Limitations and Future work

In this work, we reformulate neural operators within a $(d + 1)$ -dimensional framework, in which latent functions live on $D_x \times D_p$ and are evolved by operators acting jointly over the physical and auxiliary coordinates. Building on this formulation, we propose the Schrödingerised Kernel Neural Operator (SKNO) as one Fourier-based instantiation with residual linear blocks and basis-diversified auxiliary evolution along D_p . Across more than ten benchmarks, SKNO attains the lowest relative L_2 error among the compared baselines under the reported settings. Its resolution robustness, zero-shot temporal generalization, and controlled ablation studies support the empirical effectiveness of the

proposed $(d + 1)$ -dimensional design. Taken together, these results suggest that auxiliary-domain operator evolution is a promising alternative to brute-force embedding scaling.

An important future direction is to develop quantitative criteria for comparing neural operators beyond final test error alone. Because training navigates a high-dimensional error landscape over a large set of kernel parameters, optimization trajectories and attained local minima can vary substantially across architectures. While existing models (e.g., [36, 14, 65]) and related theoretical analyses (e.g., [28, 31, 12]) provide approximation-oriented results, they offer limited insight into which minima are selected by practical training. Future work will therefore study how different aggregation designs and propagation parameters affect neighborhood-averaged training and test errors around optimizer-attained solutions.

References

- [1] David Adalsteinsson and James A Sethian. A fast level set method for propagating interfaces. *Journal of computational physics*, 118(2):269–277, 1995.
- [2] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 2002.
- [3] Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K Gupta. Clifford neural layers for pde modeling. In *the 11th International Conference on Learning Representations*, 2023.
- [4] Brian J Cantwell. *Introduction to symmetry analysis*. Cambridge University Press, 2002.
- [5] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6):631–640, 2024.
- [6] Shuhao Cao. Choose a transformer: Fourier or galerkin. In *Advances in Neural Information Processing Systems*, 2021.
- [7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.
- [8] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.
- [9] Francisco Chinesta, Pierre Ladeveze, and Elias Cueto. A short review on model order reduction based on proper generalized decomposition. *Archives of Computational Methods in Engineering*, 18(4):395–404, 2011.
- [10] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in Neural Information Processing Systems*, 2019.
- [11] Alexandre Ern and Jean-Luc Guermond. *Theory and practice of finite elements*, volume 159. Springer, 2004.
- [12] Takashi Furuya, Koichi Taniguchi, and Satoshi Okuda. Quantitative approximation for neural operators in nonlinear parabolic equations. In *the 13th International Conference on Learning Representations*, 2025.
- [13] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. In *the 10th International Conference on Learning Representations*, 2022.
- [14] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

- [16] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly journal of the royal meteorological society*, 146(730):1999–2049, 2020.
- [17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [18] Shi Jin and Nana Liu. Analog quantum simulation of partial differential equations. *Quantum Science and Technology*, 9:035047, 2024.
- [19] Shi Jin and Stanley Osher. A level set method for the computation of multi-valued solutions to quasi-linear hyperbolic pde’s and hamilton-jacobi equations. *Communications in Mathematical Sciences*, 1(1):575–591, 2003.
- [20] Shi Jin, Nana Liu, and Yue Yu. Quantum simulation of partial differential equations via schrodingerisation: technical details. *ArXiv*, abs/2212.14703, 2022.
- [21] Shi Jin, Nana Liu, and Yue Yu. Quantum simulation of partial differential equations: Applications and detailed analysis. *Physical Review A*, 108:032603, 2023.
- [22] Shi Jin, Nana Liu, and Yue Yu. Quantum simulation of partial differential equations via schrödingerization. *Physical Review Letters*, 133(23):230602, 2024.
- [23] Armand Kassaï Koupaï, Jorge Mifsut Benet, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Boosting generalization in parametric pde neural solvers through adaptive conditioning. *Advances in Neural Information Processing Systems*, 2024.
- [24] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435, 2021.
- [25] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *the 3rd International Conference on Learning Representations*, 2015.
- [26] Georgios Kissas, Jacob H Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, George J Pappas, and Paris Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.
- [27] Jean Kossaifi, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized fourier neural operator for high-resolution pdes. *Transactions on Machine Learning Research*, 2024.
- [28] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22(290): 1–76, 2021.
- [29] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24:89:1–89:97, 2023.
- [30] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5): 987–1000, 1998.
- [31] Samuel Lanthaler, Zongyi Li, and Andrew M Stuart. Nonlocality and nonlinearity implies universality in operator learning. *Constructive Approximation*, pages 1–43, 2025.
- [32] Zhihao Li, Haoze Song, Di Xiao, Zhilu Lai, and Wei Wang. Harnessing scale and physics: A multi-graph neural operator framework for pdes on arbitrary geometries. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 729–740, 2025.
- [33] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations’ operator learning. *Transactions on Machine Learning Research*, 2023.

- [34] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. In *the 8th International Conference on Learning Representations, ICLR 2020 Workshop: Integration of Deep Neural Models and Differential Equations*, 2020.
- [35] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in Neural Information Processing Systems*, 2020.
- [36] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *the 9th International Conference on Learning Representations*, 2021.
- [37] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.
- [38] Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 2024.
- [39] Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. *ArXiv*, abs/2402.16845, 2024.
- [40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *the 7th International Conference on Learning Representations*, 2019.
- [41] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.
- [42] David J Lucia, Philip S Beran, and Walter A Silva. Reduced-order modeling: new approaches for computational physics. *Progress in aerospace sciences*, 40(1-2):51–117, 2004.
- [43] Yves Meyer. *Wavelets and operators*. Number 37. Cambridge university press, 1992.
- [44] Abdel-Rahim Mezidi, Jordan Patracone, Saverio Salzo, Amaury Habrard, Massimiliano Pontil, Rémi Emonet, and Marc Sebban. A bregman proximal viewpoint on neural operators. In *the 13th International Conference on Learning Representations*, 2025.
- [45] Octavi Obiols-Sales, Abhinav Vishnu, Nicholas Malaya, and Aparna Chandramowlishwaran. Cfdnet: A deep learning-based accelerator for fluid simulations. In *Proceedings of the 34th ACM International Conference on Supercomputing*, pages 1–12, 2020.
- [46] Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 2024.
- [47] Stanley Osher and Ronald P Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational physics*, 169(2):463–502, 2001.
- [48] Stanley Osher, Ronald Fedkiw, and K Piechor. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.*, 57(3):B15–B15, 2004.
- [49] Jiagang Qu, Weihua Cai, and Yijun Zhao. Learning time-dependent pdes with a linear and nonlinear separate convolutional neural network. *Journal of Computational Physics*, 453: 110928, 2022.
- [50] Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023.

- [51] Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes. *Advances in Neural Information Processing Systems*, 2023.
- [52] Bogdan Raonic, Roberto Molinaro, Tobias Rohner, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.
- [53] Meer Mehran Rashid, Tanu Pittie, Souvik Chakraborty, and NM Anoop Krishnan. Learning the stress-strain fields in digital composites using fourier neural operator. *Iscience*, 25(11), 2022.
- [54] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [55] Jin Young Shin, Jae Yong Lee, and Hyung Ju Hwang. Pseudo-differential neural operator: Generalized fourier neural operator for learning solution operators of partial differential equations. *ArXiv*, abs/2201.11967, 2022.
- [56] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 2022.
- [57] Karn Tiwari, NM Krishnan, et al. Cono: Complex neural operator for continuous dynamical systems. *ArXiv*, abs/2310.02094, 2023.
- [58] Karn Tiwari, Niladri Dutta, NM Anoop Krishnan, and Prathosh AP. Latent mamba operator for partial differential equations. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.
- [59] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *the 11st International Conference on Learning Representations*, 2023.
- [60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [61] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.
- [62] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.
- [63] Eugene Wigner. On the quantum correction for thermodynamic equilibrium. *Physical review*, 40(5):749, 1932.
- [64] Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional pdes with latent spectral models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.
- [65] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.
- [66] Hao Wu, Kangyu Weng, Shuyi Zhou, Xiaomeng Huang, and Wei Xiong. Neural manifold operators for learning the evolution of physical dynamics. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3356–3366, 2024.
- [67] Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved Operator Learning by Orthogonal Attention. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

- [68] Enrui Zhang, Adar Kahana, Alena Kopaničáková, Eli Turkel, Rishikesh Ranade, Jay Pathak, and George Em Karniadakis. Blending neural operators and relaxation methods in pde numerical solvers. *Nature Machine Intelligence*, pages 1–11, 2024.

A Statement on LLM Usage

In this paper, we used a large language model (LLM) to provide suggestions on word choice, grammar, and logical consistency. While some sentences were lightly rephrased by the LLM, all scientific content remains original and authored by us.

B Ethics and Reproducibility Statements

This work does not involve human subjects, personal data, or other sensitive information. All experiments were conducted on publicly available or synthetically generated datasets. To ensure reproducibility, we provide detailed descriptions of the experimental setup, including model architectures, training procedures, and evaluation details. The code and datasets used in this study will be made publicly accessible upon publication. For readers, the code implementations and related configurations are included in the supplementary materials.

C Table of Notations

A table of key notations is given in Table 12.

D Model, Experiment Settings and Results

Loss Denote the parameterized neural operator as $\mathcal{G}_\theta$, we aim to approximate the solution operator $\mathcal{G}$ in Eq. 6 by optimizing the model parameters $\theta \in \Theta$ through relative L_2 loss below:

$$\min_{\theta \in \Theta} \mathcal{L}(\theta) := \min_{\theta \in \Theta} \frac{1}{S} \sum_{s=1}^S \left[\frac{\|\mathcal{G}_\theta[a_s] - u_s\|_2}{\|u_s\|_2} \right] \quad (10)$$

$$= \min_{\theta \in \Theta} \frac{1}{S} \sum_{s=1}^S \left[\left(\frac{\sum_j (\mathcal{G}_\theta[a_s](x_j) - u_s(x_j))^2}{\sum_j (u_s(x_j))^2} \right)^{\frac{1}{2}} \right], \quad (11)$$

where Θ , S , and $u_s = u_s(x)$ denote the parameter space, the number of function samples, and the i -th output function, respectively.

Benchmarks and Results Benchmarks and the corresponding experimental results are summarized below:

1. 1D Heat Equation

Consider the 1D Heat Equation as follows:

$$\partial_t u = c \partial_{xx} u, \quad (12)$$

where $c = 0.05$ as in [55]. We generate 1200 samples (1000 for the training set, and 100 for testing) in a denser grid with size $2^{10} = 1024$, and down-sample them to size $2^8 = 256$. Our goal is to predict solution $u_1 = u(x, t = 1)$ from $u_0 = u(x, t = 0)$. Results are shown in Table 2.

2. 1D Advection Equation

Consider 1D Advection Equation:

$$\partial_t u = \beta \partial_x u, \quad (13)$$

where β is set with 1. We use the data from [56] with 2000 samples of $u_0 = u(x, t = 0)$. Our goal is to predict the next 10 time steps from the last 10 of the solution $u(x, t)$. The number of training and testing samples is 1000 and 200, respectively. Results are shown in Table 2

3. 1D Burgers Equation

The 1D Burger's Equation on the unit torus is formed as [36]:

$$\partial_t u + u \partial_x u = \nu \partial_{xx} u, \quad (14)$$

where $\nu = 0.1$ with periodic boundary conditions. Our goal is to build a solution operator to predict the solution $u(x)$ at $t = 1$. The training and testing grid size is set with $2^7 = 128$. Results are listed in Table 1.

4. 2D Gray-Scott System

Consider a pair of coupled reaction-diffusion PDEs as follows:

$$\begin{aligned}\partial_t u &= D_u \Delta u - uv^2 + F(1 - u), \\ \partial_t v &= D_v \Delta v - uv^2 - (F + k)v,\end{aligned}\tag{15}$$

where $D_u = 0.2097$, $D_v = 0.105$ denote the diffusion coefficients respectively for u, v . And $F = 0.03$, $k = 0.062$ are the reaction parameters. We follow the data generation in [23] to make a training set with 200 samples of u, v evolution and a testing set with 20 in a 32×32 size grid with periodic boundary conditions. Our goal is to predict the next 10 time steps from the last 10 of u, v . See Table 1 for results.

5. 2D Shallow-Water Equations

The 2D shallow-water equations govern the evolution of fluid flows under the assumption that horizontal length scales are much larger than the vertical depth. They can be expressed as the following system of hyperbolic PDEs:

$$\partial_t h + \nabla h \mathbf{u} = 0,\tag{16}$$

$$\partial_t h \mathbf{u} + \nabla \left(\mathbf{u}^2 h + \frac{1}{2} g_r h^2 \right) = -g_r h \nabla b,\tag{17}$$

where $h = h(x, t)$ denotes the fluid depth, $\mathbf{u} = \mathbf{u}(x, t)$ is the 2D velocity field, g_r is the (reduced) gravitational acceleration, $b = b(x)$ represents the bathymetry characterizing the bottom topography. For this benchmark, we employ the dataset from [56], which is generated under Neumann boundary conditions. To assess resolution-invariant stability during training, models are trained on 500 samples with resolutions randomly drawn from $\{32, 64, 128\}$ and evaluated on 300 test samples at the highest resolution 128. The prediction task is to forecast the water depth over the next 10 time steps, given the previous 10. See Table 17 for results.

6. 2D Darcy Flow

The 2D Darcy Flow is a second-order elliptic equation as follows:

$$-\nabla \cdot (a \nabla u) = f,\tag{18}$$

where $f = 1$. Our goal is to approximate the mapping operator from the coefficient distribution function $a(x)$ to the solution function $u(x)$ with zero Dirichlet boundary conditions [36]. The results are based on models trained on 1000 training samples, which are down-sampled to a size of (85×85) , and tested on 100 samples. Results are listed in Table 13.

7. 2D Incompressible Navier-Stokes Equations

The 2D incompressible Navier-Stokes (NS) Equations in vorticity form on a unit torus are shown below:

$$\partial_t \omega + \mathbf{u} \cdot \nabla \omega - \nu \Delta \omega = f,\tag{19}$$

$$\nabla \cdot \mathbf{u} = 0.\tag{20}$$

Here $\nu = 1e-5$ and $f = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$. Our goal is to predict the next 10 time steps from the last 10 ($d_a = d_u = 10$) of the solution $w(x, t)$ with periodic boundary conditions on the 2D grid with size 64×64 [36]. All models are trained on 1000 training samples and tested on 200 test samples. Results are listed in Table 1 and Table 14.

8. 3D Compressible Navier-Stokes Equations

The 3D compressible NS equations describe fluid dynamics with density changes, given by:

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0,\tag{21}$$

$$\rho(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \Delta \mathbf{v} + (\zeta + \eta/3) \nabla(\nabla \cdot \mathbf{v}),\tag{22}$$

$$\partial_t \left[\epsilon + \frac{\rho v^2}{2} \right] + \nabla \cdot \left[\left(\epsilon + p + \frac{\rho v^2}{2} \right) \mathbf{v} - \mathbf{v} \cdot \sigma' \right] = 0,\tag{23}$$

where $\mathbf{v} = \mathbf{v}(x, t)$, $\rho = \rho(x, t)$, and $p = p(x, t)$ denote the velocity vector field, fluid density (mass per unit volume), and pressure, respectively. We consider the challenging transonic flow

benchmark from [56], with shear and bulk viscosity coefficients $\eta = \zeta = 1 \times 10^{-8}$, and internal energy density $\epsilon = \frac{3p}{2}$. The viscous stress tensor $\sigma' = \sigma'(\mathbf{v}; \eta, \zeta)$ is determined by the velocity field under the Newtonian fluid assumption. Our objective is to predict the velocity, density, and pressure fields three time steps ahead, given the previous three steps ($d_a = d_u = 3 \times 5 = 15$). The simulations are performed on a downsampled 3D grid of size $32 \times 32 \times 32$ with periodic boundary conditions. The initial conditions consist of turbulent velocity fields with uniform mass density and pressure. All benchmark models are trained on 500 samples and evaluated on 100 test samples. See Table 1 for results.

9. 3D Rayleigh-Taylor Instability

The Rayleigh–Taylor (RT) instability occurs when a heavier fluid overlies a lighter one under the influence of gravity, which is extremely sensitive to perturbations of the interface and governed by the following PDEs:

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0, \quad (24)$$

$$\partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{g}, \quad (25)$$

$$\nabla \cdot \mathbf{u} = -\kappa \nabla \cdot \left(\frac{\nabla \rho}{\rho} \right). \quad (26)$$

where $\rho = \rho(x, t)$ and $\mathbf{u} = \mathbf{u}(x, t)$ denote the density and the velocity vector field, respectively. p is the pressure, $\boldsymbol{\tau} = \boldsymbol{\tau}(\mathbf{u}, \rho; \nu)$ is the eviatoric stress tensor with kinematic viscosity ν , $\mathbf{g}$ represents the gravity, and κ is the coefficient of molecular diffusivity. A key dimensionless parameter in the RT simulations is the Atwood number $A = (\rho_h - \rho_l) / (\rho_h + \rho_l)$, which quantifies the density contrast between the two fluids. For example, miscible fluids with greater density contrasts exhibit stronger instabilities. In this work, we adopt the dataset with $A = 0.25$ provided in [46] to train a solution operator $\text{vec}(\rho, \mathbf{u}; t = t_0) \mapsto \text{vec}(\rho, \mathbf{u}; t = t_0 + 20)$ which evolves the system from rest states (with small perturbations) to mixed states after 20 time steps. The models are trained on 220 mapping pairs and tested on 40 pairs from distinct trajectories, using a downsampled grid of size $32 \times 32 \times 32$. Horizontal periodic boundary conditions and impermeable vertical walls are imposed. See Table 1 for results.

10. ERA5 Wind Field Prediction

To compare the baseline models in a global context of spatio-temporal dynamics, we evaluate their performance on predicting the next-hour ERA5 global wind field [16]. All models are trained using data from the previous four days to predict the next-hour wind field at a grid size of 512×512 , and are tested on data from a new day with the same grid size. See Table 18 for results.

We emphasize the broader learning difficulty spectrum of the benchmarks we selected. As shown in Figure 7 which illustrates the spectral decay of truncation modes, our chosen benchmarks 6 and 7 contain significantly more high-frequency modes than simpler benchmarks such as 3 or other incompressible NS benchmarks with $\nu > 1 \times 10^{-4}$. To further evaluate the capability of these efficient and lightweight neural operators in capturing complex non-linear dynamics in 3D space, we include two additional, extremely challenging benchmarks: 8 and 9. Moreover, we also compare the performance of all baseline models in the rapid prediction of global wind fields, providing a comprehensive evaluation across both highly nonlinear and large-scale spatio-temporal dynamics.

Baselines The principle of our experiment configurations for different baselines is to keep the best configuration provided in their papers or codes, while ensuring fairness and not exploding the *Nvidia GeForce RTX 4090* GPU memory (limited to 24564MiB). Detailed configurations of all baselines in different experiments are shown in Table 6. And some explanation notes are listed below:

- Mainly two types of Positional Features (Pos. Feat.), absolute coordinates (Abs.) and reference grid (Ref.), are concatenated into input signals: (1) FNO [36] concatenates the input function value $a \in \mathbb{R}^{d_a}$ with the normalized absolute position coordinate $x \in \mathbb{R}^d$, usually scaled between 0 and 1. (2) Transolver [65], on the other hand, uses an advanced version which constructs a coarse reference grid and computes Euclidean distances from each input point to each point in this reference grid, concatenating these relative positions $x_{ref} \in \mathbb{R}^{(d_{ref})^d}$. In our work, we utilize the same absolute positional features as FNO to ensure a fair comparison.

- Since Transolver employs the multi-head in its implementation, it creates sub-grid with a equal size from the complete one.
- For Transolver, the optimizers and scheduler type is AdamW [40] and OneCycleLR, while for FNO and SKNO are Adam [25] and Cosine Annealing. All the learning rates are initially set to 0.001. More configurations can be found in the code provided in our supplementary materials.

Zero-shot Super-resolution To assess the resolution-invariant inference capability of SKNO, we evaluate the model on three benchmarks: 1D Burgers’ equation (trained on a grid of size 128), 2D Darcy Flow (trained on a grid of size 85×85), and ERA5 Global Wind Field Prediction (trained on a grid of size 32×32). As shown in Table 15 and Table 16, SKNO maintains stability and accuracy even when tested on extremely fine grids, including 8192 , $(421)^2$, and $(512)^2$. All experiments are conducted on a single RTX 4090 GPU, consistent with the setup used in other sections.

Averaging Neural Operator on D_p In experiments governed by non-linear equations, we utilize $F_p^{-1} \tilde{A} F_p + B$, where $\tilde{A}$ and B are both matrices with diagonally enhanced initialization. Such kind of implementation can be thought as an Averaging Neural Operator (ANO) [31] along p , which governs the universal approximation of the PDE operator on D_p .

Table 6: Complete recommended training configurations

Model	Epochs	Batch Size	# Modes	OR Slices	# Layers	# Pos. Feat. / Type	# Sub-grid $\times$ size on D_p
1D Heat (Linear Block)							
Transolver	500	20	4		1	N.A.	1×4
FNO	500	20	1		1	N.A.	1×4
SKNO	500	20	1		1	N.A.	1×4
1D Advection (Linear Block)							
Transolver	500	4	8		1	N.A.	1×16
FNO	500	20	8		1	N.A.	1×16
SKNO	500	20	8		1	N.A.	1×16
1D Burgers							
Transolver	500	4	64		8	64 / Ref.	8×32
FNO	500	20	16		4	1 / Abs.	1×64
SKNO	500	20	16		4	1 / Abs.	1×64
2D Gray-Scott							
Transolver	500	2	32		8	$8^2 = 64$ / Ref.	8×32
FNO	500	10	12		4	2 / Abs.	1×24
SKNO	500	10	12		4	2 / Abs.	1×24
2D Shallow-Water							
Transolver	100	2	32		8	$8^2 = 64$ / Ref.	8×32
FNO	100	20	8		4	2 / Abs.	1×24
SKNO	100	20	8		4	2 / Abs.	1×24
2D Darcy Flow							
Transolver	500	4	64		8	$8^2 = 64$ / Ref.	8×16
FNO	500	20	12		4	2 / Abs.	1×32
SKNO	500	20	12		4+1	2 / Abs.	1×32
2D Incompressible Navier-Stokes							
Transolver	500	2	32		8	$8^2 = 64$ / Ref.	8×32
FNO	8000	20	12		4	2 / Abs.	1×24
SKNO	3000	20	12		4+1	2 / Abs.	1×24
3D Compressible Navier-Stokes							
Transolver	100	2	32		8	$8^3 = 512$ / Ref.	8×32
FNO	100	10	16		4	3 / Abs.	1×32
SKNO	100	10	16		4	3 / Abs.	1×32
3D Rayleigh-Taylor Instability							
Transolver	250	2	32		8	$8^3 = 512$ / Ref.	8×32
FNO	250	10	16		4	3 / Abs.	1×32
SKNO	250	10	16		4+1	3 / Abs.	1×32
ERA5 Wind Field Prediction							
Transolver	50	2	64		4	$8^2 = 64$ / Ref.	8×8
FNO	50	2	10		4	2 / Abs.	1×20
SKNO	50	2	10		4	2 / Abs.	1×20

E Related Work

Adding one dimension to solve PDEs Unlike traditional methods [42, 9] that rely on dimension reduction techniques, such as symmetry analysis [4], to represent equations within their original domain, approaches like the level set method [47, 48, 1, 19] and the recent Schrödingerisation method [18, 21, 22] propose extending original PDEs to one more higher dimensional domains to create a pipe for functions to evolve. Similar "adding one dimension" ideas have also been used in signal processing like wavelets [43] which employ scale and shift dimensions to represent original signals, and in quantum mechanics like the Wigner functions [63] which utilize a single function displaying the probability (quasi-)distribution in the phase space.

In this paper, Schrödingerisation is used only as design inspiration for introducing an auxiliary function coordinate and Fourier-domain evolution along that coordinate. We do not claim that Schrödingerisation itself provides a classical numerical acceleration mechanism for the nonlinear well-posed PDE benchmarks considered here. The theoretical and empirical claims concern the resulting NO architecture.

Neural Networks for PDEs In scenarios where PDEs are unknown or training data is limited, supervised learning methods are employed to approximate solution operators, which recently shows their capacity to universally represent solution operators for partial differential equations (PDEs) [17, 28, 41, 30]. Existing approaches mainly fall into two paradigms: Common Neural Networks and Neural Operators.

Common NNs approximate solution operators working on a fixed grid size, which maps discretized input to output signals using architectures such as MLPs [15, 7], CNNs [49, 3]. For example, [45, 24] utilizes a typical CNN structure to map the input to the output image, which is composed of the solution of numerical solvers.

Neural operators (NOs) are designed to map between function spaces, enabling generalization across varying input discretizations without retraining [29, 61]. One of the most popular NO definitions is the kernel integral operator [36, 52, 5]. Early kernel function implementations employed graph-based message-passing networks [34, 35], where dense edge connections encoded these kernels. However, these methods faced scalability challenges due to long training times and inefficiencies in modeling interactions across multiple spatial and temporal scales. Transformer-based models [26], similar to graphs, also encounter limitations in computational efficiency due to quadratic time complexity. To address these challenges, tricky computation strategies on kernels have been proposed. Galerkin-based methods [6, 11] employ linear projections to parameterize kernels, while [67] introduces a decomposition constrained by positive semi-definite assumptions. However, such methods suffer from the problems of reduced expressiveness due to the absence of non-linear activations like softmax. Recent SOTA model transolver [65] proposes a physics-attention module to approximate the kernel function with the aid of a self-attention-based aggregation on flattened data. While these graph or transformer-based advanced methods try to escape from the heavy quadratic complexity by tricky implementations on the kernel functions originally defined in [34, 36], they still pay the similar computational price for their neglect of the embedding designs. While Mamba-based methods such as [58] can further reduce the computational cost over the domain after aggregation, the similar embedding scaling needs remain unaddressed. Instead, empirically designed fully connected or multi-head architectures are typically retained by default to model the evolution along the auxiliary dimension.

Additionally, Fourier Neural Operators (FNO) [36] efficiently implement spectral kernel representations by global convolutions via the Fast Fourier Transform (FFT). While training fast, it also faces the representation bottleneck like other NO methods because of the unaligned evolution in embedding spaces. Subsequent works [13, 62, 59, 27, 44] take efforts to extend the Fourier Transform to other analytically well-explored transformations. For example, Complex Neural Operator [57] generalizes the Fourier Transformation with an extra learnable rotation parameter to model the d -dimensional kernel integration.

There is another NO definition: DeepONet [41] employs a branch-trunk architecture inspired by the universal operator approximation theorem [8]. These representations are combined through a dot product for the embeddings composed with functional outputs to predict the solution values. Recent advancements, such as integrating numerical solvers with DeepONet [68], have improved

convergence by capturing multi-frequency components, enhancing performance on complex PDE systems. However, its general-purpose design may limit its effectiveness in scenarios requiring high accuracy and PDE-aligned structural representations.

Embedding augmentation and auxiliary coordinates. Augmenting a latent representation is a common design principle in continuous-depth models and operator learning. For example, augmented neural ODEs [10] enlarge an embedding vector so that continuous flows can represent maps that may be difficult to realize in the original size. Similarly, most neural operators lift pointwise input function values $a(x)$ into an enlarged embedding before evolving them to improve approximation. In our formulation, the auxiliary variable p is treated as a function coordinate where the latent scalar function $v(x, p)$ on $D_x \times D_p$ evolves, and the update is parameterized by operators acting over this product domain. This makes the auxiliary direction available for kernel design.

Demonstrations of the Schrödingerisation We demonstrate how the d -dimensional linear PDEs are transformed into their $d + 1$ -dimensional versions in Schrödingerisation after introducing an auxiliary dimension p :

Example 1 (Heat Equation [20, 21]). Consider the initial-value problem of the heat equation:

$$\partial_t u = \partial_{xx} u, \quad (d \text{ dimensional heat equation}) \quad (27)$$

$$u(x, t = 0) = u_0, \quad (d \text{ dimensional input}) \quad (28)$$

where $u = u(x, t)$, $x = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$, $t \geq 0$. Let $d = 1$, which can be easily generalized to arbitrary d . Our goal is to construct a $d + 1$ dimensional pipe that evolves the initial condition $u_0 = u(x, t = 0)$ to the solution $u_1 = u(x, t = 1)$.

Step 1: Warped Phase Transformation. Define a transformation from $u(x, t)$ to $v(x, p, t)$ by introducing an auxiliary dimension $p \in (-\infty, \infty)$:

$$v(x, p, t = 0) = v_0(x, p) = e^{-|p|} u_0. \quad (29)$$

Step 2: $d + 1$ Dimensional Phase-space Heat Equation. Combine Eq.27, 28 and 29, obtain:

$$\partial_t v(x, p, t) = -\partial_p \partial_{xx} v(x, p, t). \quad (30)$$

Step 3: Recovering to d Dimensional Solution. After evolving $v_0 \mapsto v_1 := v(x, p, t = 1)$ by Eq.30, Recover u_1 by integrating along p :

$$u_1(x) = \int_{-\infty}^{\infty} \chi(p) v_1(x, p) dp, \quad (31)$$

where $\chi(p)$ has multiple choices and depends on the introduced transformation form in Step 1. Here $\chi(p)$ can be formed as the step function or delta function.

For Eq.29, if taking fourier transform on p ($\mathcal{F}_p : p \rightarrow \eta$), one gets a group of uncoupled Schrödinger equations (operators), over all η ; If taking fourier transform on x ($\mathcal{F}_x : x \rightarrow \xi$), one gets a group of uncoupled convection equations (operators), over all ξ . Generally for $v(t)$, discrete signals on x and p from $v(x, p, t)$, one gets a Hamiltonian system by taking the discrete fourier transform (DFT) on p . For Eq.29, the Hamiltonian system can be represented by further taking DFT on x :

$$\frac{d}{dt} \hat{v}(t) = i(D_\mu^2 \otimes D_\mu) \hat{v}(t), \quad (32)$$

where $\hat{v}(t)$ is derived after taking DFT for $v(t)$ on p and x , and the same notation $D_\mu = \text{diag}(\mu_{-N_p/2}, \dots, \mu_{N_p/2})$, which denotes a diagonal matrix with N_p grid points in one dimension, is employed since little confusion will arise. Here, the diagonal coefficient complex matrix $i(D_\mu^2 \otimes D_\mu)$ is from taking DFTs on the x and p domain for $-\partial_p \partial_{xx}$ in Eq.29.

Example 2 (Advection Equation and Other Linear PDEs). Similar process and result on advection equation as Example 1 with introducing corresponding warped phase transformation $v_0 = \sin(p) u_0$, $p \in [-\pi, \pi]$. For more discussion of other PDE types, refer to [21].

F Properties of SKNO

We highlight several properties of SKNO below.

Complexity To analyze computational complexity across general dimensional settings (1D, 2D, 3D, ...), let N_x denote the number of grid points in the spatial domain, N_p the grid size along the auxiliary dimension, and k_x the number of aggregated slices/nodes (used in Transolver). The complexity comparison is summarized in Table 7.

Table 7: Complexity comparison of different kernel implementations.

Models	Complexity
Transolver	$\mathcal{O}[N_p(N_x(k_x + N_p) + k_x^2)]$ ($k_x \geq 32$, $N_p = 128$ or 256)
FNO	$\mathcal{O}[N_p(N_x \log(N_x) + N_p k_x)]$ ($k_x \leq 16$, $N_p \leq 32$)
SKNO (Ours)	$\mathcal{O}[N_p(N_x(\log N_x + \log N_p) + N_p k_x)]$ ($k_x \leq 16$, $N_p \leq 32$)

Table 8: Complexity comparison of different lifting implementations with positional encoding. (Assume all choose one linear layer)

Models	Complexity
Transolver	$\mathcal{O}[N_p(d_a + (d_{ref})^d)]$ ($d_{ref} = 8$, $N_p = 128$ or 256)
FNO	$\mathcal{O}[N_p(d_a + d)]$ ($N_p \leq 32$)
SKNO (Ours)	$\mathcal{O}[N_p(d_a + d)]$ ($N_p \leq 32$)

Consider an extreme but practically relevant case in high-accuracy numerical simulations: a 3D field with grid resolution $((2)^9)^3 = (512)^3$ and $N_p = 32$ (while Transolver typically requires $N_p = 128$ for stable performance). Even with the minimal $k_x = 32$, Transolver’s kernel implementation with a large linear constant ($k_x + N_p > 128$) leads to a computational cost much higher than SKNO ($\log N_x = 27$) or FNO ($\log N_x N_p = 27 + 5 = 32$).

In lower dimensions like 1D or 2D cases, where N_x is exponentially smaller, our kernel retains the linear-complexity advantages while achieving superior performance with significantly smaller scale constants compared to the SOTA Transolver (under the log benefit from Fast Fourier Transform). Here, we report the FLOPs of the kernel integration on representative experiments in Table 9. Notably, when the positional encodings are removed in Table 8, the prediction error of Transolver increases dramatically (from $6e - 3$ to $6e - 1$ in relative L_2 error on Burgers), whereas the other two models maintain nearly the same accuracy level.

Table 9: Computational cost comparison (FLOPs) of the kernel integration and relative rate (normalized by FNO).

Metric \ Model	FNO	Transolver	SKNO
<i>2D Navier–Stokes</i>			
FLOPs	1.16×10^8	1.19×10^9	2.12×10^8
Rate	1.000	10.292	1.831
<i>3D Rayleigh–Taylor Instability</i>			
FLOPs	8.98×10^8	9.00×10^9	1.59×10^9
Rate	1.000	10.029	1.774

Spectral Operator along p Since our SKNO adopts a spectral implementation, we primarily compare it to the FNO [36] as a reference framework. This choice also facilitates generalization to other spectral models like [13, 38]. To avoid notational clutter, we begin with the $d = 1$ case, noting that the following arguments extend straightforwardly to higher dimensions.

Theorem F.1 (Neural Operator along the Auxiliary Dimension). *Let $v(x, p)$ be taken from a compact set V in the appropriate Sobolev space so that its truncated Fourier representatives*

$$\hat{v}_i := F_{k_x}^x [P_{k_x}^x v](\hat{x}_i, \mathbf{p}) \in \hat{V}_i \subset \mathbb{C}^{N_p}, \quad \tilde{v}_i := F^p [R_{k_p}^p (F_{k_x}^x [P_{k_x}^x v])(\hat{x}_i, \tilde{\mathbf{p}})] \in \tilde{V}_i \subset \mathbb{C}^{N_p},$$

are compact sets in a finite coefficient space. Then

1. *Approximating an operator $\mathcal{M}$ is equivalent (via the Fourier-conjugate decomposition) to approximating the conjugate mapping(s) $\{\widehat{\mathcal{M}}_i\}_{i=1}^{k_x}$ (and, after also Fourier-transforming in p , the conjugates $\{\widetilde{\mathcal{M}}_i\}$).*
2. *The forward DFT/frequency-projection and the inverse discrete transform admit neural operator approximation on the spectral domain (See Lemma 7 and Lemma 8 in [28], hence the operator learning problem reduces to a finite-dimensional neural-network approximation on the compact coefficient sets $\hat{V}_i, \tilde{V}_i$.*

In particular, treating the auxiliary p -dimension via an additional DFT (as SKNO does) simply produces an enlarged conjugate mapping; the same approximation strategy (as Lemma 7 + Lemma 8 + finite NN on compact coefficients) applies.

Note. $\widehat{\mathcal{M}}_i : \hat{V}_i \subset \mathbb{R}^{2N_p} \rightarrow \mathbb{R}^{2N_p}$ for $i = 1, \dots, k_x$ and $\widetilde{\mathcal{M}}_i : \tilde{V}_i \subset \mathbb{R}^{2N_p} \rightarrow \mathbb{R}^{2N_p}$. Usually, we take $k_p = N_p$ and the $\frac{\text{length}(\mathbf{p}) - k_p}{\text{length}(\mathbf{p})}$ random dropout $R_{k_p}^p = I^p$.

Proof. By the Fourier-conjugate decomposition (Eq. (12) in [28]), the operator can be written as

$$\begin{aligned} \mathcal{M} &= (F_{k_x}^x)^{-1} \circ \widehat{\mathcal{M}}_{k_x} \circ F_{k_x}^x \circ P_{k_x}^x \\ &= (F_{k_x}^x)^{-1} \circ (F^p)^{-1} \circ \widetilde{\mathcal{M}}_{k_x} \circ F^p \circ F_{k_x}^x \circ P_{k_x}^x. \end{aligned} \quad (33)$$

Here, the second line follows from additionally applying a DFT in the auxiliary variable p .

The mapping in Eq.33 $F^p F_{k_x}^x [P_{k_x}^x (\cdot)]$ sends the compact input family V into a compact subset of Fourier coefficient space. Lemma 7 and Lemma 8 of [28] guarantee that the left-hand and right-hand parts of both $\widehat{\mathcal{M}}_{k_x}$ and $\widetilde{\mathcal{M}}_{k_x}$ can be approximated within the desired accuracy.

Thus, the problem reduces to approximating the conjugate operators $\widehat{\mathcal{M}}_{k_x}$ or $\widetilde{\mathcal{M}}_{k_x}$, both of which are continuous maps defined on compact subsets of $\mathbb{R}^{2N_p}$. By the universal approximation theorem for neural networks [2, 17], any such continuous map can be approximated to arbitrary accuracy by a finite neural network. Hence, there exists finite NNs that approximate the conjugate mappings on $\hat{V}_i$ (or $\tilde{V}_i$) within the desired accuracy. From the Theorem 5 in [28], SKNO universally approximates any d -dimensional operator $\mathcal{G}$.

Next, we try to explain why the auxiliary update used by SKNO can be more parameter-efficient than duplicating a standard same-basis channel-mixing branch.

Proposition F.2 (Complementary-basis enlargement under a finite auxiliary-operator budget). *Fix one spatial Fourier mode and write the discretized auxiliary profile as $z \in \mathbb{C}^{N_p}$. Let $\Phi \in \mathbb{C}^{N_p \times N_p}$ be an invertible basis transform along the auxiliary coordinate p ; in SKNO, $\Phi = F_p$ is the DFT. Let $\mathcal{S} \subseteq \mathbb{C}^{N_p \times N_p}$ denote the linear subspace realizable by one finite-budget auxiliary branch, e.g., a diagonal, block, truncated, Hermitian-constrained, or otherwise structured parameterization. Define the same-basis two-branch class and the complementary-basis class by*

$$\mathcal{H}_{B+B} := \{S_1 + S_2 : S_1, S_2 \in \mathcal{S}\}, \quad (34)$$

$$\mathcal{H}_{A+B} := \{S + \Phi^{-1} T \Phi : S, T \in \mathcal{S}\}. \quad (35)$$

If $\mathcal{S}$ is a linear subspace, then

$$\mathcal{H}_{B+B} = \mathcal{S}. \quad (36)$$

In contrast,

$$\dim(\mathcal{H}_{A+B}) = \dim(\mathcal{S}) + \dim(\Phi^{-1} \mathcal{S} \Phi) - \dim(\mathcal{S} \cap \Phi^{-1} \mathcal{S} \Phi). \quad (37)$$

Therefore, whenever the transformed subspace is not contained in the original one, i.e., $\Phi^{-1} \mathcal{S} \Phi \not\subseteq \mathcal{S}$, we have

$$\mathcal{H}_{B+B} \subsetneq \mathcal{H}_{A+B}. \quad (38)$$

Table 10: Capacity Efficiency on 2D Gray-Scott. Comparison of SKNO against FNO under systematically varied embedding sizes ($1\times$ to $4\times$), parallel stacking (2/3/4-Stack), and branch architecture (B+B vs. A+B), evaluated over 6 random seeds with 500 training epochs. SKNO (A+B) achieves the lowest error while using fewer parameters and fewer FLOPs than FNO at any tested scaling configuration. FNO scaling is non-monotonic, peaking at $2\times$ embedding size and degrading beyond. SKNO(B+B), which replaces the Fourier-domain branch along p -dim with a duplicate standard branch at matched parameter budget, fails to improve over FNO, confirming the architectural origin of SKNO’s gains.

Model	# Params	Rel. L_2 Err. $\pm$ Std.	FLOPs
<i>FNO with varying embedding size</i>			
FNO	1,337,450	$2.169\text{e-}2 \pm 3.35\text{e-}3$	940,318,720
FNO (+1 emb.)	1,451,177	$2.042\text{e-}2 \pm 1.24\text{e-}3$	998,184,960
FNO ($1.33\times$ emb.)	2,377,186	$1.869\text{e-}2 \pm 2.21\text{e-}3$	1,445,109,760
FNO ($1.5\times$ emb.)	3,008,414	$1.834\text{e-}2 \pm 2.67\text{e-}3$	1,733,386,240
FNO ($2\times$ emb.)	5,347,538	$1.685\text{e-}2 \pm 3.73\text{e-}3$	2,741,739,520
FNO ($3\times$ emb.)	12,030,266	$1.952\text{e-}2 \pm 1.83\text{e-}3$	5,404,303,360
FNO ($4\times$ emb.)	21,385,634	$1.839\text{e-}2 \pm 1.10\text{e-}3$	8,928,010,240
<i>Parallel stacking in the same basis</i>			
2-Stack FNO	2,685,962	$2.532\text{e-}2 \pm 4.29\text{e-}3$	2,328,371,200
3-Stack FNO	4,027,370	$2.214\text{e-}2 \pm 2.26\text{e-}3$	3,429,376,000
4-Stack FNO	5,368,778	$2.063\text{e-}2 \pm 3.30\text{e-}3$	4,530,380,800
<i>SKNO branch ablation</i>			
SKNO (B+B)	2,690,762	$2.447\text{e-}2 \pm 9.65\text{e-}4$	2,521,047,040
SKNO (A+B)	2,141,844	$1.256\text{e-}2 \pm 1.14\text{e-}3$	2,112,141,236

Proof. Since $\mathcal{S}$ is closed under addition, $S_1 + S_2 \in \mathcal{S}$ for all $S_1, S_2 \in \mathcal{S}$, and every element of $\mathcal{S}$ is obtained by choosing $S_2 = 0$. Hence $\mathcal{H}_{B+B} = \mathcal{S}$. The expression for $\dim(\mathcal{H}_{A+B})$ is the standard dimension formula for the sum of two linear subspaces, applied to $\mathcal{S}$ and $\Phi^{-1}\mathcal{S}\Phi$. If $\Phi^{-1}\mathcal{S}\Phi \not\subseteq \mathcal{S}$, then there exists $T \in \mathcal{S}$ such that $\Phi^{-1}T\Phi \notin \mathcal{S}$; this element belongs to $\mathcal{H}_{A+B}$ by choosing $S = 0$, but it does not belong to $\mathcal{H}_{B+B} = \mathcal{S}$. Thus the inclusion is strict. $\square$

Corollary F.3 (Finite-budget approximation error). *For any target auxiliary linear update L^* ,*

$$\inf_{H \in \mathcal{H}_{A+B}} \|L^* - H\|_F \leq \inf_{H \in \mathcal{H}_{B+B}} \|L^* - H\|_F.$$

$$\inf_{H \in \mathcal{H}_{A+B}} \|L^* - H\|_F \leq \inf_{H \in \mathcal{H}_{B+B}} \|L^* - H\|_F, \quad (39)$$

with strict improvement whenever L^ has a component in $\Phi^{-1}\mathcal{S}\Phi$ outside $\mathcal{S}$. Thus, adding a same-basis branch can be redundant, whereas adding a conjugated branch can enlarge the reachable finite-budget operator family. This explains the role of SKNO’s $F_p^{-1}\tilde{A}F_p + B$ update: the raw- p and Fourier- p terms provide two coordinate views of the same auxiliary profile.*

Remark F.4 (Unitary stability). If Φ is unitary, as for the DFT, then

$$\|\Phi^{-1}T\Phi\|_2 = \|T\|_2, \quad \|\Phi^{-1}T\Phi\|_F = \|T\|_F.$$

Thus the complementary branch changes represented directions without increasing operator norms.

Sparsity. Due to the limited network width, N_p should be finite and as small as possible. Benefiting from the PDE-aligned propagator design, SKNO constructs a sparser dictionary on the output side of the $d + 1$ dimensional propagation pipes that is more efficient in capturing the energy of the solutions than FNO.

For a clear and fair comparison, we consider SKNO without the differential operator and the baseline FNO. The lifting and recovering operators in both models are chosen as unbiased linear layers without positional encoding. Under the same configurations, we compare the dictionaries of SKNO and FNO at the last output linear layer in Figure 5, which illustrates the sparsity advantage of the SKNO dictionary.

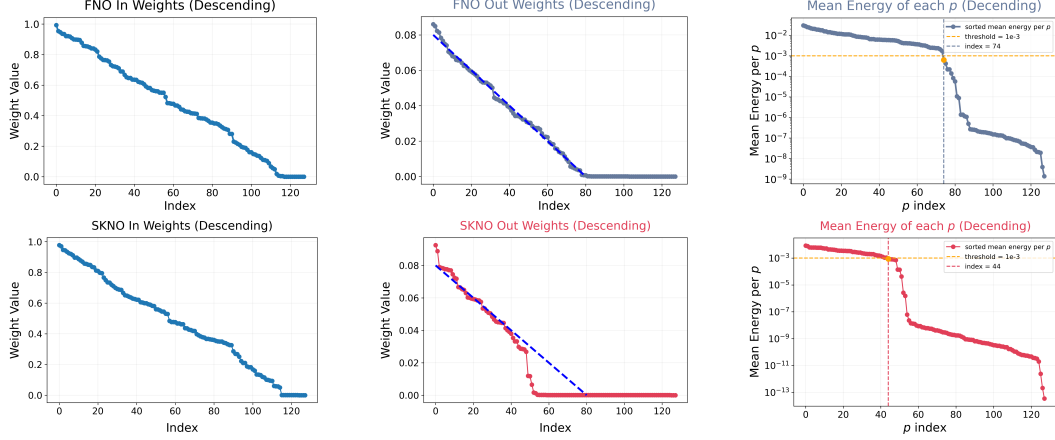


Figure 5: Dictionary comparison of FNO and SKNO. We train both models with a sufficiently large recovering budget N_p on the Burgers benchmark and visualize the absolute values of the input/output weights, together with the mean energy of the dictionary of pattern fields. While the input weights of FNO (top left) and SKNO (bottom left) exhibit nearly identical distributions, SKNO yields a noticeably sparser structure in its output weights (bottom middle) and in the mean energy of each pattern field (bottom right) compared to FNO. The yellow horizontal line denotes a mean-energy threshold of 10^{-3} , where modes below this threshold contribute only marginally to the final prediction. For a complete visualization of the output-weight and pattern-field pairs in the dictionary, see Figure 16.

Beyond the experiments, we also provide a formal proof in a finite-dimensional matrix–vector setting, which aligns well with the neural implementation, to clarify the efficient energy capture of SKNO. The extension to function spaces amounts to working in the corresponding infinite-dimensional setting.

Lemma F.5 (Preserved Energy after Top- r Truncation). *For a general matrix–vector multiplication $\mathbf{u} = G\mathbf{y}$ where $G \in \mathbb{R}^{N_x \times N_p}$ and $N_x \geq N_p$, if we perform the top- r approximation G_r using the singular value decomposition (SVD) of G , the energy preserved in the prediction $G_r\mathbf{y}$ is*

$$\|G_r\mathbf{y}\|_2^2 = \sum_{m=1}^r \lambda_m^2 \langle \mathbf{y}, \phi_m \rangle^2, \quad r \leq N_x, \quad (40)$$

where $G_r\mathbf{y} = \Psi_r \Lambda_r \Phi_r^T \mathbf{y} = \sum_{m=1}^r \lambda_m \langle \mathbf{y}, \phi_m \rangle \psi_m$, and ψ_m , ϕ_m , and λ_m are the m -th left singular vector, right singular vector, and singular value, respectively.

Proof. This follows directly from $G_r\mathbf{y}$ representing with the orthonormal basis $\{\psi_m\}$ of $\text{range}(r)$.

Proposition F.6 (Conditional energy-capture dominance under limited width). *Assume the two models use the same lifting and recovery operators and that, under the considered width budget, the realizable last-layer dictionary class of FNO is contained in that of SKNO,*

$$\mathcal{H}_{\text{FNO}}(r) \subseteq \mathcal{H}_{\text{SKNO}}(r).$$

Define the best achievable top- r captured energy

$$E_M^*(r) := \sup_{V_L^{(M)} \in \mathcal{H}_M(r)} \frac{\|(V_L^{(M)})_r \mathbf{x}^{(M)}\|_2^2}{\|\mathbf{u}\|_2^2}.$$

Then

$$E^{(\text{SKNO})}(r) \geq E^{(\text{FNO})}(r),$$

and therefore the corresponding best rank- r truncation error of SKNO is no larger than that of FNO.

Note. (1) The length N_x in the following discussion is obtained by flattening the d -dimensional tensor while keeping the sequential order. (2) Without loss of generality, we discuss the case $d_u = 1$.

Proof. Denote the vector $\mathbf{u} \in \mathbb{R}^{N_x}$ discretized on D_x , the vector $\boldsymbol{\chi}^{(M)} \in \mathbb{R}^{N_p}$ discretized on N_p , the matrix $V_L^{(M)} \in \mathbb{R}^{N_x \times N_p}$ where $N_x \geq N_p$. We focus on the last output (linear) layer in neural operators M. If N_p is sufficiently large, we have

$$\begin{aligned}\mathbf{u} &= V_L^{(M)} \boldsymbol{\chi}^{(M)} \\ &= \Psi_L^{(M)} \Lambda_L^{(M)} (\Phi_L^{(M)})^T \boldsymbol{\chi}^{(M)},\end{aligned}\tag{41}$$

with singular values $\lambda_1^{(M)} \geq \lambda_2^{(M)} \geq \dots \geq \lambda_{N_x}^{(M)}$. The matrices $\Psi_L^{(M)} = (\psi_1^{(M)}, \psi_2^{(M)}, \dots, \psi_{N_x}^{(M)})$ and $\Phi_L^{(M)} = (\phi_1^{(M)}, \phi_2^{(M)}, \dots, \phi_{N_p}^{(M)})$ are composed with their column vectors $\{\psi_m\}_{m=1}^{N_x}$ and $\{\phi_m\}_{m=1}^{N_p}$, respectively.

In practice, we apply the r -truncation on N_p as the optimal approximation under the metric in Eq. 10. Utilizing the rank- r truncated SVD, we have

$$\begin{aligned}(V_L^{(M)})_r &:= (\Psi_L^{(M)})_r (\Lambda_L^{(M)})_r (\Phi_L^{(M)})_r^T \\ &= \sum_{m=1}^r \lambda_m^{(M)} \psi_m^{(M)} (\phi_m^{(M)})^T, \quad r \leq N_x.\end{aligned}\tag{42}$$

From Lemma F.5, the (relative) energy of u captured by the top- r SVD modes of model M is

$$\begin{aligned}E^{(M)}(r) &:= \frac{\|(V_L^{(M)})_r \boldsymbol{\chi}^{(M)}\|_2^2}{\|\mathbf{u}\|_2^2} \\ &= \frac{\sum_{m=1}^r (\lambda_m^{(M)})^2 \langle \boldsymbol{\chi}^{(M)}, \phi_m^{(M)} \rangle^2}{\sum_{m=1}^{N_x} (\lambda_m^{(M)})^2 \langle \boldsymbol{\chi}^{(M)}, \phi_m^{(M)} \rangle^2}.\end{aligned}\tag{43}$$

Since singular values are non-negative, the sequence $E^{(M)}(r)$ is non-decreasing with respect to r for each model M. And the truncation error is

$$\|\mathbf{u} - (V_L^{(M)})_r \boldsymbol{\chi}^{(M)}\|_2 = \|\mathbf{u}\|_2 \sqrt{1 - E^{(M)}(r)}, \quad r = 1, \dots, N_x.\tag{44}$$

Specifically, the error difference between FNO and SKNO is

$$\begin{aligned}\epsilon(r) &= \frac{\|\mathbf{u} - (V_L^{(\text{FNO})})_r \boldsymbol{\chi}\|_2 - \|\mathbf{u} - (V_L^{(\text{SKNO})})_r \boldsymbol{\chi}\|_2}{\|\mathbf{u}\|_2} \\ &:= \sqrt{1 - E^{(\text{FNO})}(r)} - \sqrt{1 - E^{(\text{SKNO})}(r)}.\end{aligned}\tag{45}$$

From Eq.43, $\epsilon(r) \rightarrow 0$ as $r \rightarrow N_p$.

Let $\mathcal{H}^{(M)}$ denotes the set of $V_L^{(M)}$ that can be realized by NO M under the same width budget r . We have

$$E^{(M)}(r) := \sup_{V_L^{(M)} \in \mathcal{H}_M} \frac{\|(V_L^{(M)})_r \boldsymbol{\chi}^{(M)}\|_2^2}{\|\mathbf{u}\|_2^2}, \quad M \in \{\text{FNO}, \text{SKNO}\}, \quad r \leq N_p.\tag{46}$$

Since $\mathcal{H}_{\text{FNO}} \subset \mathcal{H}_{\text{SKNO}}$, the supremum over $\mathcal{H}_{\text{SKNO}}$ cannot be smaller than the supremum over $\mathcal{H}_{\text{FNO}}$, which implies

$$E^{(\text{SKNO})}(r) \geq E^{(\text{FNO})}(r), \quad \forall r \leq N_p.\tag{47}$$

Adjoint Backward Propagation Under the squared loss function like Eq. 10, we show the adjoint backward propagation property of SKNO.

Proposition F.7 (Discrete backward updates for SKNO auxiliary branches). *Here we show the backward updates around linear kernel integrations in SKNO. Assume $d_a = d_u = 1$ and denote the matrix multiplication on the last output layer $V_L^{(\text{SKNO})} \chi^{(\text{SKNO})}$ and the backward input residual $\mathbf{r}^{(\text{SKNO})}$, where the vector $\chi^{(\text{SKNO})} \in \mathbb{R}^{N_p}$ is discretized on N_p , and the matrix $V_L^{(\text{SKNO})} \in \mathbb{R}^{N_x \times N_p}$ where $N_x > N_p$. The backpropagation [54] update*

$$\Delta \chi^{(\text{SKNO})} = \sum_i (V_L^{(\text{SKNO})})_i^T \cdot \mathbf{r}_i^{(\text{SKNO})} \quad (48)$$

$$\text{Grad}(V_L^{(\text{SKNO})}) = \mathbf{r}^{(\text{SKNO})} \cdot (\chi^{(\text{SKNO})})^T \quad (49)$$

$$FC : (\Delta \hat{\mathbf{B}})_{ijj'} = \text{conj}(\hat{V}_{L-1})_{ij} \cdot \hat{\mathbf{r}}_i^{(\text{SKNO})} \cdot \mathbf{M}_{ij'}^{(\text{SKNO})} \cdot (\chi^{(\text{SKNO})})_{j'}^T \quad (50)$$

$$\text{Grad}(\hat{V}_{L-1}^{(\text{SKNO})})_{ij} = \sum_{j'} \hat{\mathbf{B}}_{ijj'} \cdot \hat{\mathbf{r}}_i^{(\text{SKNO})} \cdot \mathbf{M}_{ij'}^{(\text{SKNO})} \cdot (\chi^{(\text{SKNO})})_{j'}^T \quad (51)$$

$$(\Delta \hat{\mathbf{A}})_{ijj'} = \text{conj}(\hat{V}_{L-1})_{ij} \cdot \hat{\mathbf{r}}_i^{(\text{SKNO})} \cdot \mathbf{M}_{ij'}^{(\text{SKNO})} \cdot (\tilde{\chi}^{(\text{SKNO})})_{j'}^T \quad (52)$$

$$\text{Grad}(\hat{V}_{L-1}^{(\text{SKNO})})_{ij} = \sum_{j'} \hat{\mathbf{A}}_{ijj'} \cdot \hat{\mathbf{r}}_i^{(\text{SKNO})} \cdot \mathbf{M}_{ij'}^{(\text{SKNO})} \cdot (\tilde{\chi}^{(\text{SKNO})})_{j'}^T \quad (53)$$

$$\text{Diag} : (\Delta \hat{\mathbf{B}})_{ij} = \text{conj}(\hat{V}_{L-1})_{ij} \cdot \hat{\mathbf{r}}_i^{(\text{SKNO})} \cdot \mathbf{M}_{ij}^{(\text{SKNO})} \cdot (\chi^{(\text{SKNO})})_j^T \quad (54)$$

$$\text{Grad}(\hat{V}_{L-1}^{(\text{SKNO})})_{ij} = \hat{\mathbf{B}}_{ij} \cdot \hat{\mathbf{r}}_i^{(\text{SKNO})} \cdot \mathbf{M}_{ij}^{(\text{SKNO})} \cdot (\chi^{(\text{SKNO})})_j^T \quad (55)$$

$$(\Delta \hat{\mathbf{A}})_{ij} = \text{conj}(\hat{V}_{L-1})_{ij} \cdot \hat{\mathbf{r}}_i^{(\text{SKNO})} \cdot \mathbf{M}_{ij}^{(\text{SKNO})} \cdot (\tilde{\chi}^{(\text{SKNO})})_j^T \quad (56)$$

$$\text{Grad}(\hat{V}_{L-1}^{(\text{SKNO})})_{ij} = \hat{\mathbf{A}}_{ij} \cdot \hat{\mathbf{r}}_i^{(\text{SKNO})} \cdot \mathbf{M}_{ij}^{(\text{SKNO})} \cdot (\tilde{\chi}^{(\text{SKNO})})_j^T, \quad (57)$$

where the weight $\Delta \chi \in \mathbb{R}^{N_p}$, complex tensors $\Delta \hat{\mathbf{B}} \in \mathbb{C}^{N_x \times N_p \times N_p}$ or $\mathbb{C}^{N_x \times N_p}$ and $\Delta \hat{\mathbf{A}} \in \mathbb{C}^{N_x \times N_p \times N_p}$ or $\mathbb{C}^{N_x \times N_p}$ are the changing values of parameters. $\text{conj}(\cdot)$ denotes element-wise taking the conjugate value of the input. $\mathbf{M}_{ij}^{(\text{SKNO})} = \mathbf{1}[(V_L^{(\text{SKNO})})_{ij} > 0]$ where $V_L^{(\text{SKNO})} = \text{ReLU}(V_L^{(\text{SKNO})})$.

Proof. Derive these updating values by taking partial derivatives with respect to the parameters in propagators $\hat{\mathbf{B}}$ and $\hat{\mathbf{A}}$ and in the recovering operator χ . A notable point here is the conjugate gradient backpropagation path to the Fourier domain like the $\hat{\mathbf{r}}_i^{(\text{SKNO})}$ like in Eq.51.

The adjoint backward residual propagation and conjugate parameter updates shown above are directly derived from SKNO's spectral implementation. From Theorem "F.7, we can generalize the backward propagation framework for $d + 1$ -dimensional neural operators as follows

$$\Delta v_L(x, p) = \mathcal{Q}^*[r](x, p) = \chi^T(p) r(x) \quad (58)$$

$$\mathcal{K}^*[\Delta v](x, p) = \mathcal{K}[\Delta v](x, p) = \iint_{D_x \times D_p} \kappa(x, y, p, p') \Delta v(y, p') dy dp' \quad (59)$$

$$\Delta a(x) = \mathcal{P}^*[\Delta v_0](x) = \int_{D_p} \omega(p) \Delta v_0(x, p) dp, \quad (60)$$

where $r(x) := \Delta u(x) = u(x) - \mathcal{G}[a](x)$ denotes the residual between the solution $u(x)$ and the corresponding forward prediction $\mathcal{G}[a](x)$, and we define the adjoint operator of the lifting $\mathcal{P}^*$ to have a p -integral form as the recovering operator $\mathcal{Q}$, and vice versa. See Lemma F.8.

For previous d dimensional kernel integration designs, an additional matrix inversion is required when formulating the backpropagation process, whereas $d + 1$ dimensional neural operators are

allowed to share the same kernel for the forward and adjoint operators, i.e., $\mathcal{K} = \mathcal{K}^*$. When $d + 1$ dimensional neural operators converge, the backpropagation updates [54] stabilize within a PDE-specific equivalence class of parameters, up to a certain error tolerance.

Lemma F.8 (Adjoint of the recovery operator $\mathcal{Q}$). *Let $H_x := L^2(D_x)$ and $H_{x,p} := L^2(D_x \times D_p)$ with the standard L^2 inner products. Then $\mathcal{Q}^*$ is the adjoint of $\mathcal{Q}$, i.e.*

$$\langle \mathcal{Q}[v], r \rangle_{H_x} = \langle v, \mathcal{Q}^*[r] \rangle_{H_{x,p}} \quad \forall v \in H_{x,p}, r \in H_x.$$

Proof. By definition of the L^2 inner product on H_x ,

$$\begin{aligned} \langle \mathcal{Q}[v], r \rangle_{H_x} &= \int_{D_x} (\mathcal{Q}[v])(x) r(x) dx \\ &= \int_{D_x} \left(\int_{D_p} \chi(p) v(x, p) dp \right) r(x) dx \\ &= \int_{D_x} \int_{D_p} v(x, p) \chi(p) r(x) dp dx. \end{aligned} \tag{61}$$

By definition of the L^2 inner product on $H_{x,p}$,

$$\begin{aligned} \langle v, \mathcal{Q}^*[r] \rangle_{H_{x,p}} &= \int_{D_x} \int_{D_p} v(x, p) (\mathcal{Q}^*[r])(x, p) dp dx \\ &= \int_{D_x} \int_{D_p} v(x, p) \chi(p) r(x) dp dx. \end{aligned} \tag{62}$$

Thus $\langle \mathcal{Q}[v], r \rangle_{H_x} = \langle v, \mathcal{Q}^*[r] \rangle_{H_{x,p}}$ for all v, r , so $\mathcal{Q}^*$ is the adjoint of $\mathcal{Q}$. Similar proof for the adjoint between $\mathcal{P}$ and $\mathcal{P}^*$.

Entanglement Moreover, we compute the entanglement entropy values after each $d + 1$ dimensional propagation, and plot some randomly selected cases of them in Figure 6: Compared to FNO, SKNO introduces stronger perturbations on the singular spectrum after $\mathcal{L}_l[v_l]$, cooperated with effectively controlling the sparsity of the energy distribution by σ_l . In addition, the PDE-aligned design in the $\mathcal{L}_l[v_l]$ of SKNO makes it more flexible to reduce entanglement in the final linear stage further, while FNO is purely driven by the squared-loss objective to substantially perturb the $(d + 1)$ -dimensional signal without an implicit structural regularization along p , which leads to increased entanglement entropy in some cases.

G Visualization

Patterns with different scales We visualize cases to highlight the patterns with different scales of our SKNO. These subplots are plotted over all p values after the $d + 1$ dimensional evolution. Cases include the 1D Burgers, 2D Darcy Flow, and 2D incompressible Navier-Stokes benchmarks, shown in Fig. 16, Fig. 11 and Fig. 12 respectively.

Table 11: Relative L_2 and L_1 errors of Transolver on the 2D Darcy benchmark with different numbers of attention heads.

# Heads	Rel. L_2 Error	Rel. L_1 Error
1	3.035e-2	2.894e-2
2	1.848e-2	1.684e-2
4	1.007e-2	8.297e-3
8	5.853e-3	4.899e-3

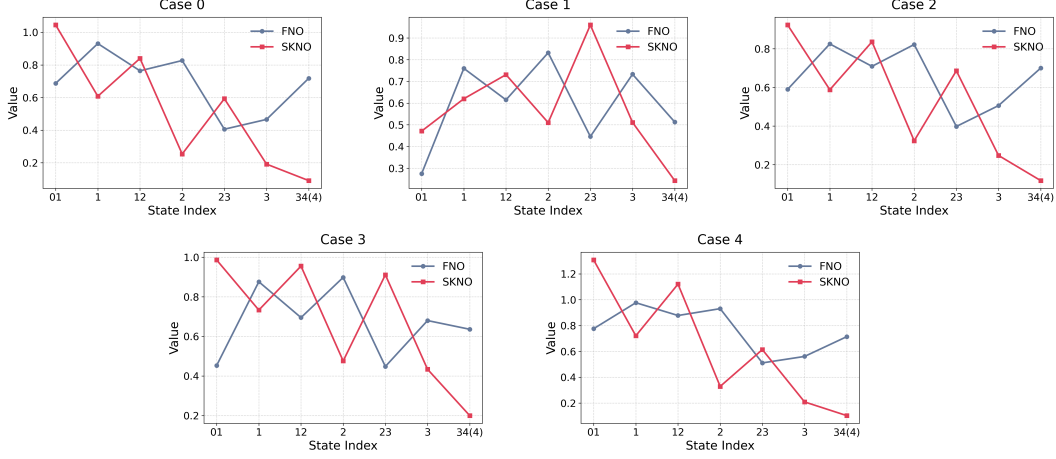


Figure 6: Entanglement entropy of intermediate layer outputs. For five randomly selected test cases from the test set, the final entanglement entropy of SKNO is consistently lower than that of FNO, indicating that the $(d+1)$ -dimensional design of SKNO yields representations that are less entangled and exhibit a more energy-concentrated distribution. On the horizontal axis, a two-digit index (e.g., "01") denotes the output after the $\mathcal{L}_0$ and before the σ_0 between v_0 and v_1 , whereas a single-digit index (e.g., "1") denotes v_1 as the output of the first propagator ($\sigma_0 \circ \mathcal{L}_0$). Note that the initial index "0" corresponds to v_0 . Under a purely linear lifting operator without positional encoding, v_0 is typically separable, so all curves effectively start from an unentangled state at the origin and evolve through the forward propagation.

Algorithm 1 Entanglement Entropy Numerical Calculation

Require: V_l , $l = 0, 1, \dots, L$.

Ensure: Entanglement Entropy S_l .

Perform singular value decomposition:

$$V_l = \Psi_l \Lambda_l (\Phi_l)^T$$

where $\Sigma_l = \text{diag}((\lambda_1)_l, (\lambda_2)_l, \dots, (\lambda_{N_p})_l)$.

Normalize Schmidt coefficients:

$$(c_m)_l \leftarrow \frac{(\lambda_m)_l}{\sqrt{\sum_{m'} (\lambda_{m'})_l^2}}.$$

Compute entanglement entropy:

$$S_l \leftarrow - \sum_m (c_m)_l^2 \log((c_m)_l^2).$$

Table 12: Table of key notations.

Notation	Meaning
Problem Setup	
$D_x \subset \mathbb{R}^d$	Original domain where functions are defined with d dimensions
$x \in D_x$	Query position of in/output functions in the d dimensional domain
$a(x) \in \mathcal{A}$	Input function
d_a	The length of concatenated input vectors
$u(x) \in \mathcal{U}$	Output function
d_u	The length of concatenated output vectors
$\mathcal{G}[\cdot]$	Solution operator
$d + 1$ dimensional NO Framework	
p	The auxiliary dimension variable composing $d + 1$ dimensional PDEs
D_p	Domain where the auxiliary variable p lives
$v(x, p)$	Transformed function in the $d + 1$ dimensional domain $D_x \times D_p$
$\mathcal{P}[\cdot]$	Operator lifts input signals into ones with grids in $D_x \times D_p$
$\mathcal{L}[\cdot]/\mathcal{L}_l[\cdot]$	The (l -th) learnable linear block capturing $d + 1$ dimensional operators
$\sigma_l(\cdot)$	The (l -th) element-wise activation function or 2-layer MLP
$\mathcal{Q}[\cdot]$	Operator recovers evolved $d + 1$ dimensional signals into outputs
Others	
$\hat{(\cdot)}$	The original function/signal represented in Fourier domain of x
$\tilde{(\cdot)}$	The original function/signal represented in Fourier domain of p
$\widehat{(\cdot)}$	The original operator/matrix represented in Fourier domain of x
$\widetilde{(\cdot)}$	The original operator/matrix represented in Fourier domain of p
N_x	The number of grid indices on D_x
N_p	The number of grid indices on D_p
k_x	The number of grid indices on the transformed domain after aggregation on D_x
k_p	The number of grid indices on the transformed domain after aggregation on D_x

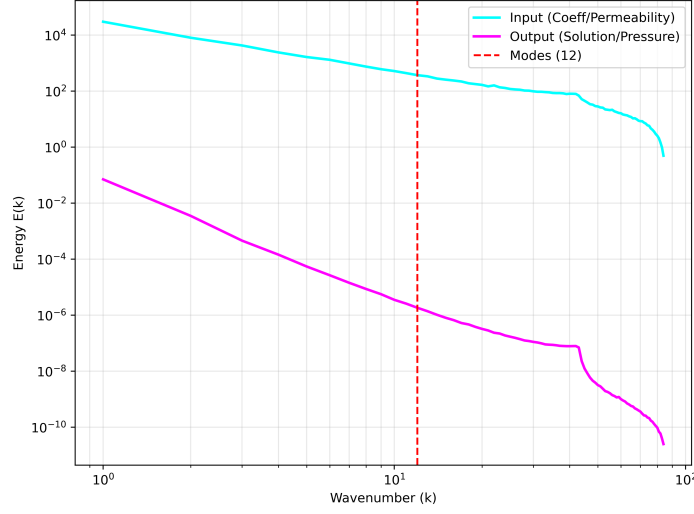
Table 13: Performance comparison on the 2D Darcy flow problem evaluating alternative implementations within d dimensional and $(d + 1)$ dimensional frameworks. The results are categorized into three distinct blocks. **Top:** Baseline d dimensional Neural Operators (NOs). **Middle:** Architectures utilizing lifting within the vector space. Specifically, *Augmented FNO* represents an FNO with expanded channel width (+1). *Hyperbolic FNO* introduces an auxiliary vector-dimension modeled as a Lorentz hyperboloid, which creates a curved feature space with constant negative curvature to capture hierarchical scale levels. *2Stack-FNO* stacks two Fourier layers on the d -dimensional domain (parallel), omitting specialized operator design along the p -dimension. **Bottom:** *SKNO-Attention* replaces the p -dimensional spectral operator in SKNO with the attention-based aggregation module from Transolver, compared against our proposed SKNO.

Model	DeepONet	GNOT	OFormer	LSM	CNO	FNO	Transolver
Rel. L_2 Err.	0.0609	0.0102	0.0121	0.0065	0.0060	0.0062	0.0058
Training Time	5.20 min	22.58 min	21.91 min	20.37 min	63.34 min	10.21 min	115.48 min

Model	<i>Augmented FNO</i>	<i>Hyperbolic FNO</i>	<i>2Stack-FNO</i>
Rel. L_2 Err.	0.0067	0.0080	0.0061
Training Time	10.82 min	15.83 min	13.35 min

Model	<i>SKNO-Attention</i>	SKNO (Ours)
Rel. L_2 Err.	0.0058	0.0055
Training Time	76.90 min	17.59 min

2D Darcy:



2D NS $\nu = 1e - 5$:

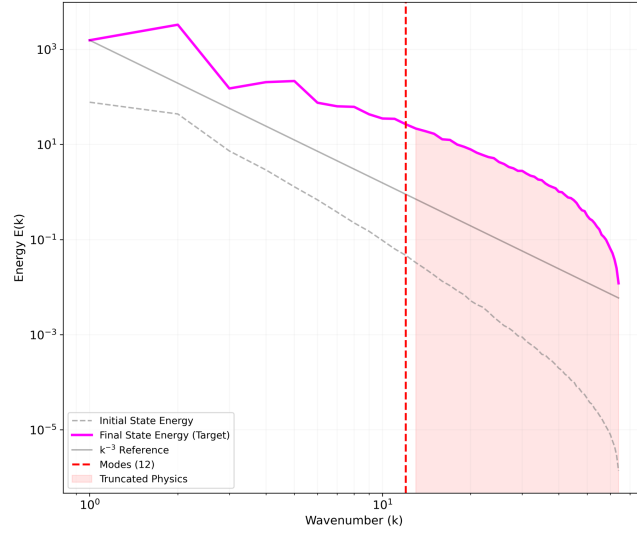


Figure 7: Energy spectral decay for the 2D Darcy flow and 2D Navier-Stokes datasets. The spectral distribution of the data provides insight into the complexity of the operator learning task. Notably, both datasets maintain substantial energy in high-frequency modes, spanning a broad and highly energetic frequency band (note that the wavenumber is plotted on a logarithmic scale). Furthermore, increased operator nonlinearity results in a more irregular spectral decay in the solution (magenta line). In both scenarios, the SKNO model demonstrates superior performance, achieving significantly lower error levels than competing models.

Table 14: Comparison of training times and relative L_2 errors for various models evaluated across six independent random seeds. FNO (time dim)*: a variant specifically designed for spatio-temporal function propagation, which preprocesses the data by repeating the spatial field to introduce a pseudo time dimension in the d -dimensional physical domain. Within the context of this study, this variant operates directly on d dimensional inputs rather than on the auxiliary embedding dimension after lifting.

Model	Seed 0		Seed 1	
	Training Time	Rel. L_2 Err.	Training Time	Rel. L_2 Err.
DeepONet [41]	874.50	5.414e-1	873.16	3.486e-1
AFNO [13]	1192.37	3.085e-1	1127.28	3.105e-1
TFNO [27]	749.36	2.270e-1	769.06	2.245e-1
U-FNO [62]	981.50	1.976e-1	986.20	1.970e-1
FFNO [59]	754.98	2.172e-1	753.16	2.304e-1
BregmanNO [44]	754.87	2.111e-1	756.86	2.202e-1
FNO [36]	843.37	1.277e-1	856.40	1.285e-1
FNO (time dim) [36] *	787.14	1.642e-1	786.21	1.663e-1
Transolver [65]	1263.33	1.002e-1	1269.94	2.086e-1
SKNO (Ours)	720.28	8.374e-2	706.68	8.726e-2

Model	Seed 2		Seed 3	
	Training Time	Rel. L_2 Err.	Training Time	Rel. L_2 Err.
DeepONet	885.19	3.534e-1	872.83	3.482e-1
AFNO	1139.20	3.081e-1	1117.37	3.062e-1
TFNO	733.09	2.263e-1	770.26	2.314e-1
U-FNO	999.03	1.949e-1	1002.90	2.000e-1
FFNO	770.89	2.283e-1	772.64	2.297e-1
BregmanNO	761.34	2.195e-1	775.92	2.212e-1
FNO	841.03	1.356e-1	849.42	1.312e-1
FNO (time dim)*	803.59	1.618e-1	789.53	1.630e-1
Transolver	1469.21	8.909e-2	1384.41	8.406e-2
SKNO (Ours)	736.79	8.355e-2	708.93	8.379e-2

Model	Seed 4		Seed 5	
	Training Time	Rel. L_2 Err.	Training Time	Rel. L_2 Err.
DeepONet	878.85	3.444e-1	880.26	3.481e-1
AFNO	1110.54	3.052e-1	1144.27	3.078e-1
TFNO	755.34	2.363e-1	768.82	2.355e-1
U-FNO	984.78	2.027e-1	997.70	1.917e-1
FFNO	769.01	2.204e-1	740.73	2.247e-1
BregmanNO	767.40	2.206e-1	755.68	2.224e-1
FNO	863.67	1.403e-1	859.44	1.370e-1
FNO (time dim)*	784.37	1.632e-1	796.33	1.655e-1
Transolver	1285.63	1.008e-1	1278.90	9.993e-2
SKNO (Ours)	738.20	8.460e-2	720.98	8.077e-2

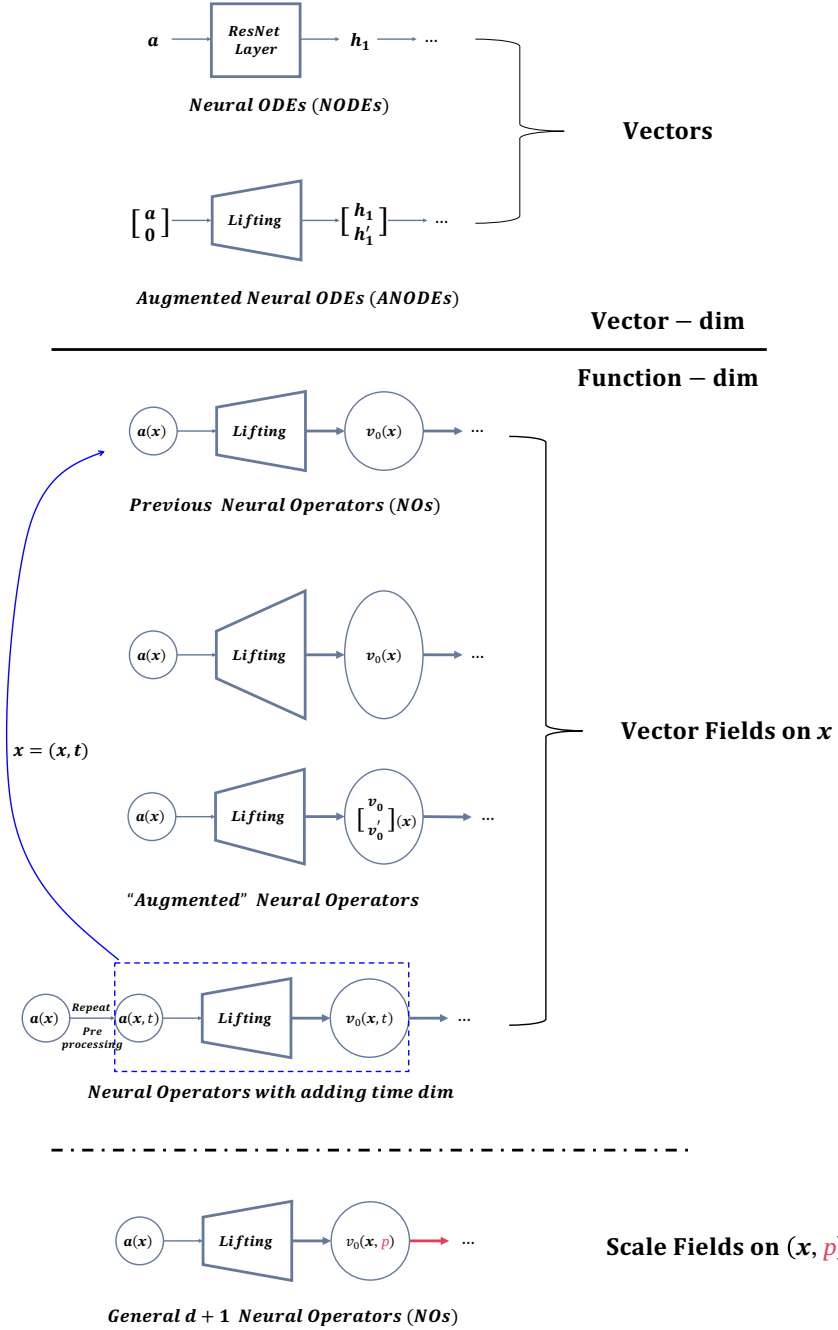


Figure 8: Architectural classifications by input and "dimension" type. **Top:** Neural ODEs (NODEs) and Augmented NODEs (ANODEs) operate on finite-dimensional vector spaces, using manual concatenation for lifting. **Middle:** Existing Neural Operators (NOs) propagate d dimensional vector fields, including variants that append a temporal dimension. **Bottom:** General $d + 1$ Neural Operators utilize an auxiliary embedding dimension to propagate $(d + 1)$ dimensional scalar fields.

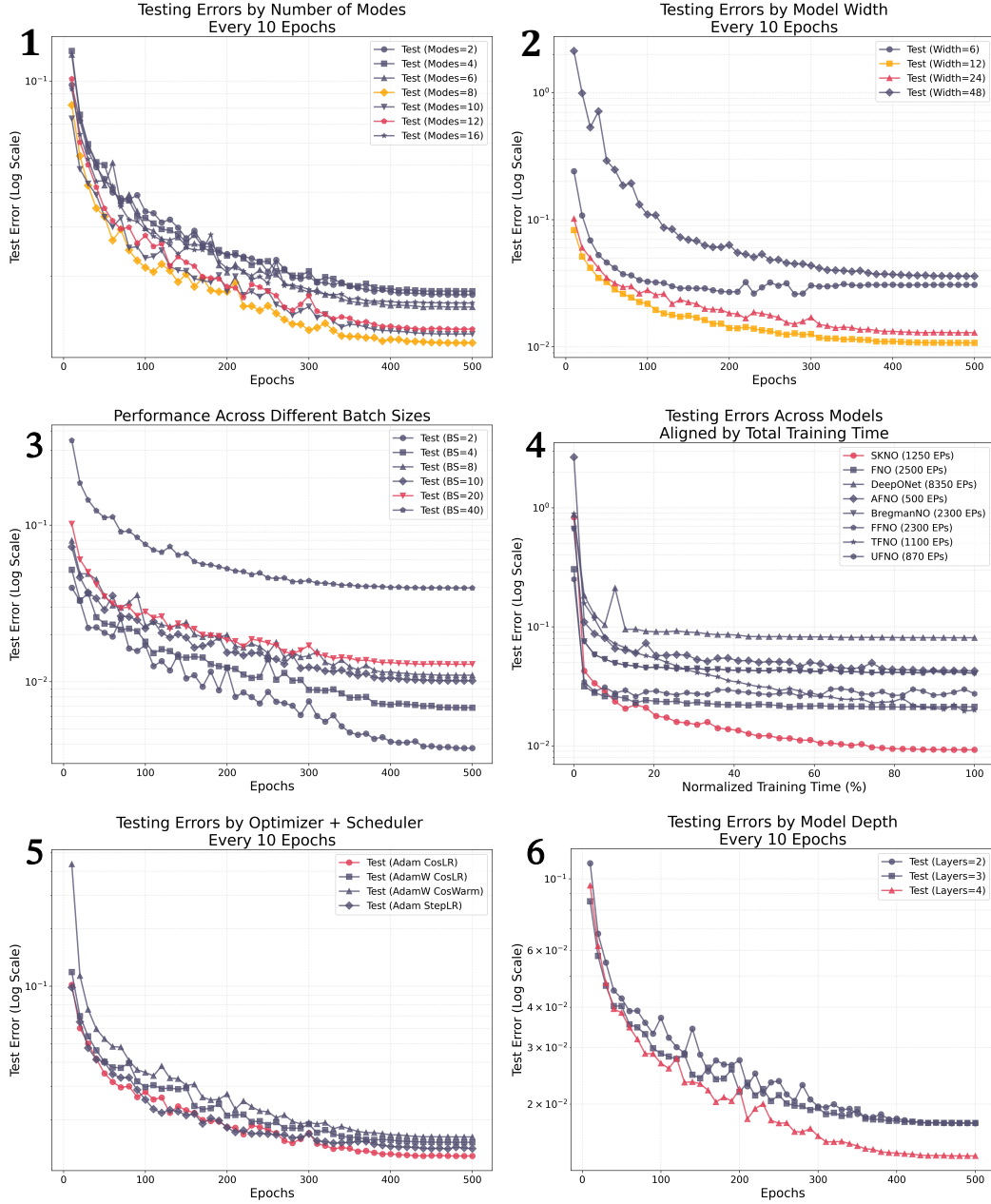


Figure 9: Ablation Studies and Performance Comparison of SKNO on the 2D Gray-Scott. All red lines represent the default configurations of SKNO used in baseline comparisons. **1** and **2**: These panels illustrate SKNO’s performance under different truncation modes in the original domain and varying sizes along p -dim (model width). The yellow lines highlight that SKNO can achieve optimal performance with a smaller number of frequency truncation modes and a more compact p -dimension size. **3**, **5**, and **6**: These plots show the influence of batch size, choice of optimizer and learning rate scheduler, and network depth on the model’s convergence and final test error. **4**: This panel displays the test error decay curves for various models aligned by the same total training time, where SKNO successfully surpasses others by above 55.79% error reduction (*Best testing performance: SKNO with $9.245e-3$ and FNO with $2.091e-2$*).

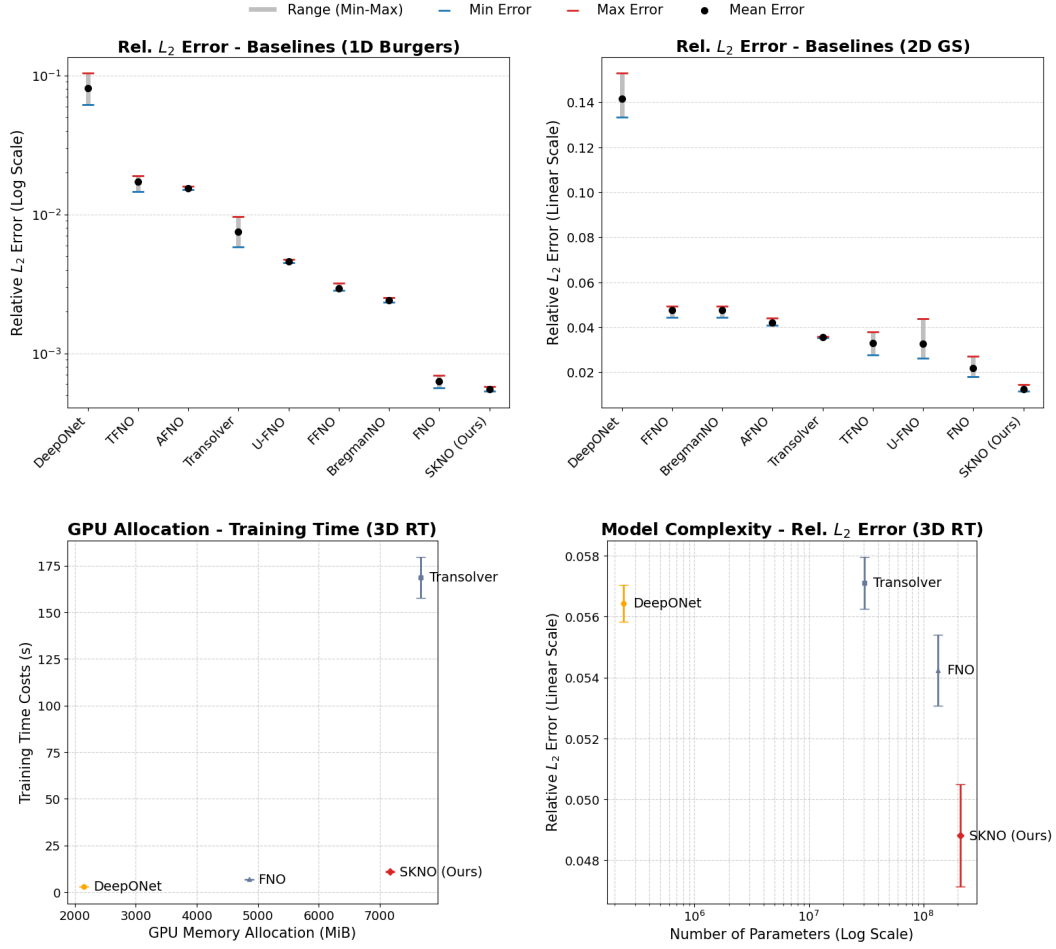


Figure 10: Visualizations of statistical performance across multiple runs with six random seeds. **Top Row:** On both the 1D Burgers and 2D Gray-Scott problems, our SKNO achieves the best performance within the same number of training epochs compared to all baselines. **Bottom Row:** On the 3D Rayleigh-Taylor Instability problem, while Transolver employs multi-head mechanisms to mitigate the explosive growth in training time caused by long embeddings, SKNO utilize in the transformed domain of the introduced auxiliary dimension, which drastically reduces training time while delivering a performance leap that remains unattainable for other baseline methods.

Table 15: Prediction errors of neural operators on the 1D Burgers equation and 2D Darcy flow across different grid sizes. The table also reports the variance of errors over grid resolutions, highlighting SKNO’s superior accuracy and stability.

1D Burgers Equation								
Grid Size	128	256	512	1024	2048	4096	8192	Variance
Transolver	6.277E-03	7.186E-03	8.058E-03	8.546E-03	8.800E-03	8.928E-03	8.993E-03	9.04472E-07
FNO	6.479E-04	6.476E-04	6.480E-04	6.477E-04	6.454E-04	6.455E-04	6.455E-04	1.33912E-12
SKNO	5.475E-04	5.489E-04	5.490E-04	5.489E-04	5.488E-04	5.487E-04	5.487E-04	2.53997E-13
2D Darcy Flow								
Grid Size	$(22)^2$	$(29)^2$	$(43)^2$	$(85)^2$	$(141)^2$	$(211)^2$	$(421)^2$	Variance
FNO	9.813E-02	7.718E-02	4.956E-02	6.165E-03	3.044E-02	3.897E-02	4.939E-02	7.83082E-04
SKNO	7.739E-02	6.302E-02	4.122E-02	5.551E-03	2.366E-02	3.271E-02	4.112E-02	4.90633E-04

Table 16: Zero-shot super-resolution evaluation on ERA5 wind field prediction. As the grid resolution increases, SKNO exhibits substantially lower growth in relative L_2 error compared to FNO, leading to a pronounced accuracy advantage on the finest grid $(512)^2$.

<i>Wind U</i>						
Grid Size	$(32)^2$	$(64)^2$	$(128)^2$	$(256)^2$	$(512)^2$	Variance
FNO	6.272E-02	6.525E-02	6.872E-02	7.097E-02	7.212E-02	1.23506E-05
SKNO	6.164E-02	6.237E-02	6.366E-02	6.445E-02	6.495E-02	1.54763E-06

<i>Wind V</i>						
Grid Size	$(32)^2$	$(64)^2$	$(128)^2$	$(256)^2$	$(512)^2$	Variance
FNO	2.165E-01	2.326E-01	2.430E-01	2.476E-01	2.498E-01	1.49635E-04
SKNO	1.969E-01	2.027E-01	2.108E-01	2.156E-01	2.181E-01	6.30471E-05

Table 17: Performance of neural operator models on 2D Shallow Water and 3D Compressible NS benchmarks. On the 2D Shallow Water, we train models on a randomly selected resolution and evaluate them at the highest resolution. —: No 3D implementation available; *: Requires predefining the resolution, here trained on the highest resolution; **: For fairness, training times are all reported on the highest resolutions. The CNO model is described in ([51]).

Models	DeepONet	CNO	FNO	Transolver	SKNO
Rel. L_2 Err.	0.0902*	0.0085*	0.0030	0.0088	0.0027
Training Time**	8.59 min	49.26 min	11.43 min	128.64 min	20.85 min

Table 18: Results for ERA5 wind field prediction.

Model	Performance	Training Time	Wind U Error	Wind V Error
FNO		0.76 min	7.064e-2	2.133e-1
Transolver		10.81 min	6.291e-2	2.007e-1
SKNO		1.75 min	6.111e-2	1.920e-1

Table 19: Comparison of different configurations with zero, one, and two local propagators. Results show that adding local aggregation layers helps signals integrate residual information more effectively with a lower level of entanglement.

Metric	w.o. Local Prop	1 Local Prop	2 Local Props
Rel. L_2 Err.	5.717	5.555	5.289
Entanglement Entropy (Before Recovering)	2.286	1.849	1.277

Table 20: Summary of benchmark dataset characteristics for the evaluated problems. The table details the number of samples in the training and testing sets, the spatial resolution or number of grid points (N_x), the input feature dimensionality (d_a), and the output field dimensionality (d_u).

BENCHMARK	TRAIN. - TEST. SAMP.	N_x	$d_a - d_u$
1D HEAT (TIME)	1000 - 100	256	1 - 1
1D ADVECTION (TIME)	1000 - 200	256	10 - 1
1D BURGERS (TIME)	1000 - 100	128	1 - 1
2D STRESS	1000 - 100	2304	1 - 3
2D STRAIN	1000 - 100	2304	1 - 3
2D DARCY	1000 - 100	7225	1 - 1
2D GRAY-SCOTT (TIME)	200 - 20	1024	20 - 2
2D SHALLOW WATER (TIME)	500 - 300	1024/4096/16384 [†]	10 - 1
2D INCOMP. NAVIER-STOKES (1E-4, TIME)	1000 - 200	4096	10 - 1
2D INCOM. NAVIER-STOKES (1E-5, TIME)	1000 - 200	4096	10 - 1
3D COM. NAVIER-STOKES (TIME)	500 - 100	32768	15 - 5
3D RAYLEIGH-TAYLOR INSTAB. (TIME)	220 - 40	32768	4 - 4
ERA5 WIND FIELD PREDICTION (TIME)	64 - 16	262144	4 - 1

[†]For the 2D Shallow Water benchmark, spatial resolutions (N_x) are randomly downsampled to grids of 32×32 , 64×64 , and 128×128 (maximum).

Table 21: Comparison of parameter counts across different models and problems. Follow the recommended configurations as the convention of NO comparisons [65].

Problem	DeepONet	Transolver	FNO	SKNO
2D Gray-Scott	86,802	11,237,698	1,337,450	2,141,844
2D Shallow-Water	16,844,624	2,175,473	599,833	971,636
2D Navier-Stokes	4,261,712	11,232,321	1,337,113	2,149,884
3D Navier-Stokes	249,989	30,339,653	134,235,877	210,028,292
3D RT	240,420	30,333,764	134,235,396	210,061,700

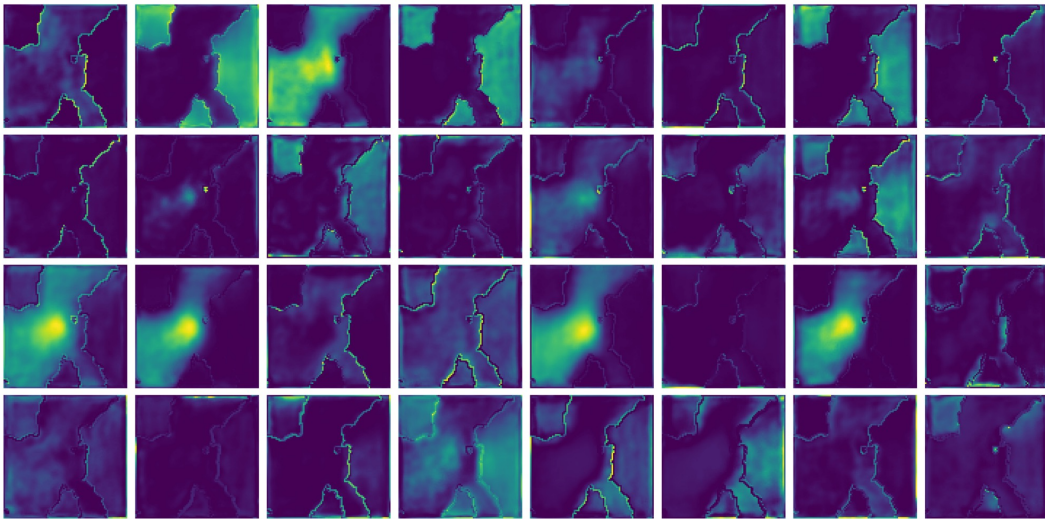


Figure 11: Pattern visualization of the 2D Darcy case. According to the discontinuities in the inputs, SKNO learns patterns at different scales.

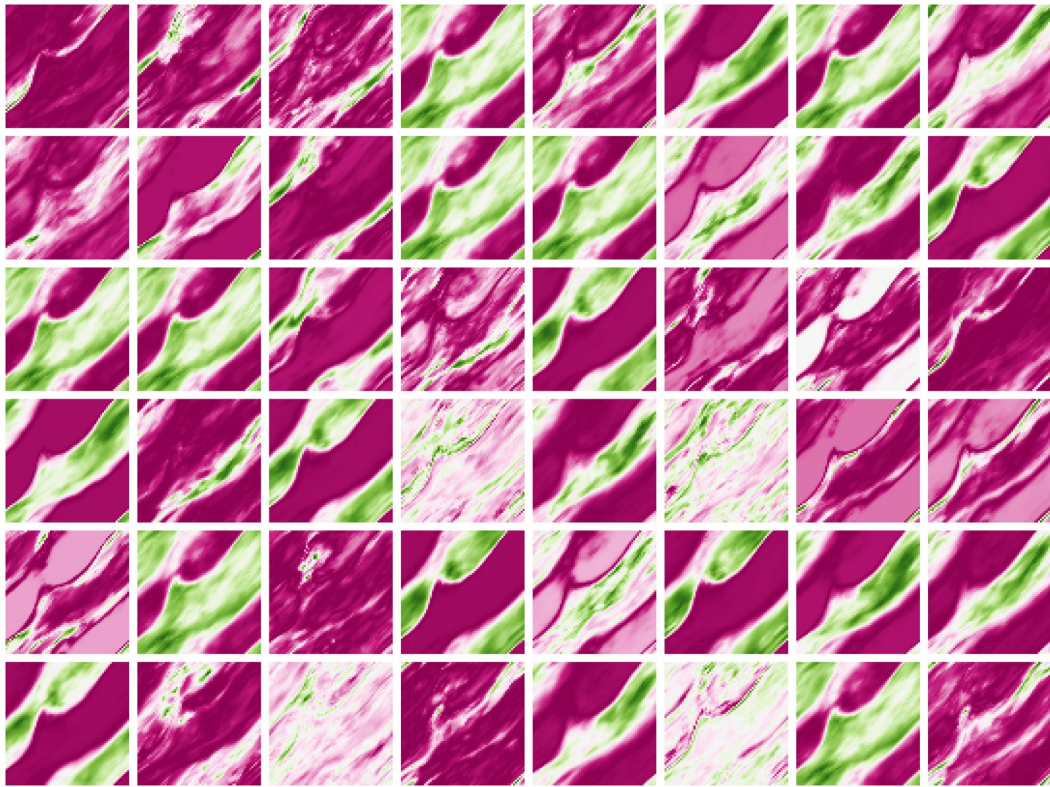


Figure 12: Pattern visualization of the 2D Navier-Stokes case.

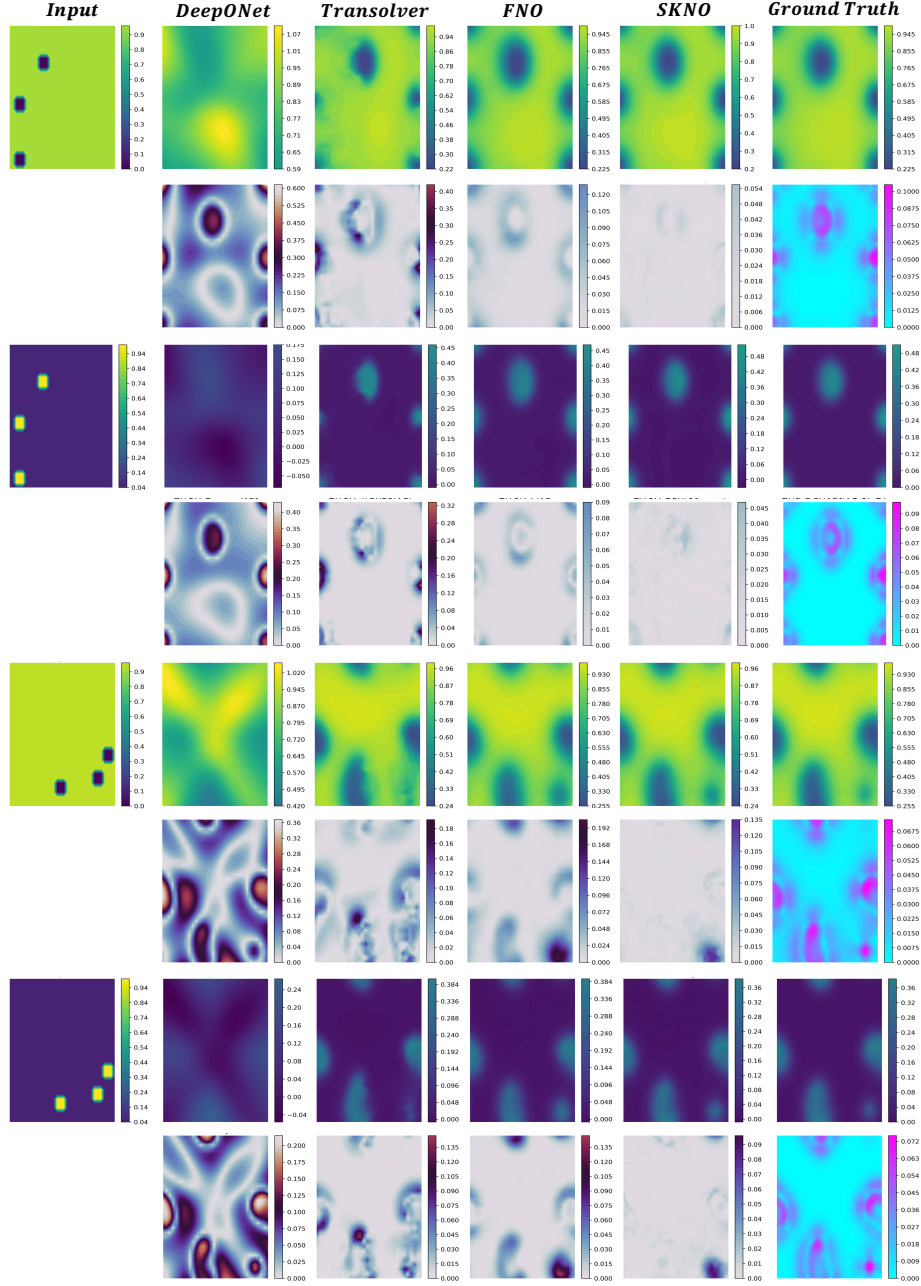


Figure 13: Visualization of model predictions for the 2D Gray-Scott dataset. The top row of each block displays the field $u(x)$, while the bottom row displays the field $v(x)$. Across all presented cases and physical fields, SKNO significantly suppresses error artifacts and reconstructs the ground truth with vastly superior precision compared to the baseline architectures.

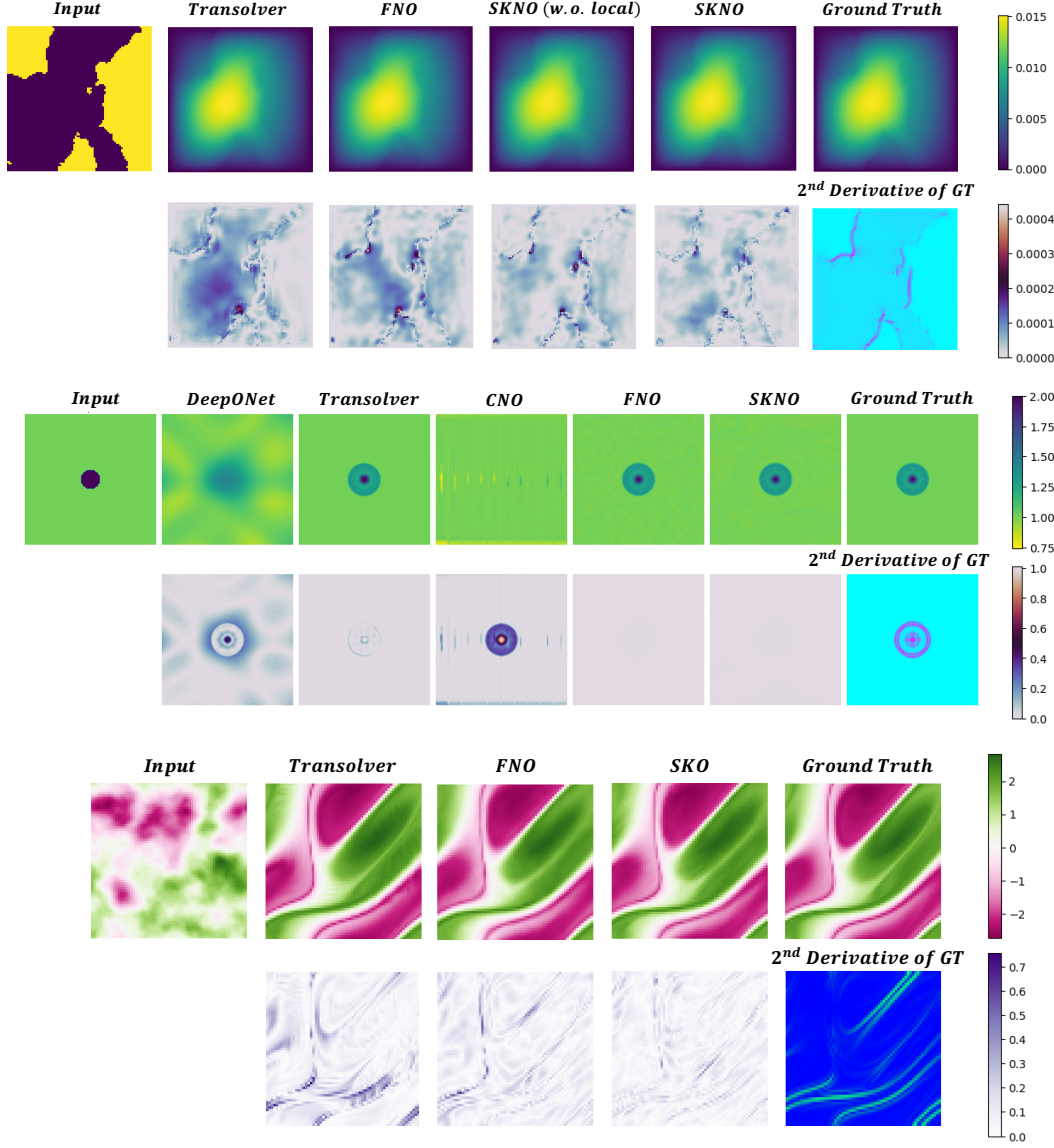


Figure 14: Visualization of predictions, errors, and the second derivative of the ground truth for 2D Darcy flow, 2D Shallow-Water equations (using randomly mixed training data), and 2D Navier-Stokes (NS) equations. In each block, the top row displays the input field, predictions from various baseline models alongside our proposed method (SKNO), and the ground truth. The bottom row presents the corresponding error visualizations for each model's prediction, as well as the second derivative of the ground truth. Overall, these comparisons illustrate the consistently low error of SKNO on physical field mapping operators across both microscopic and macroscopic scales.

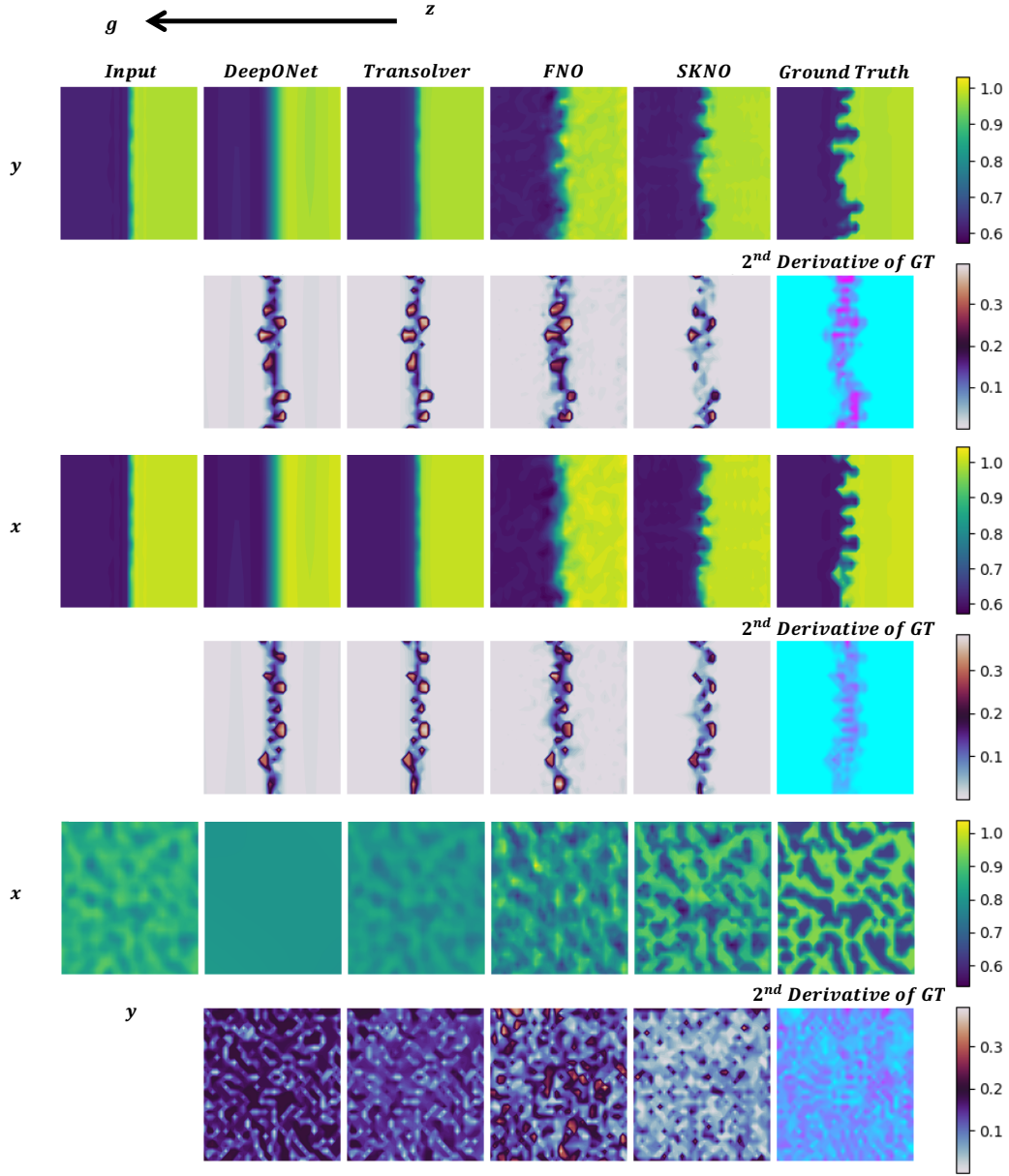
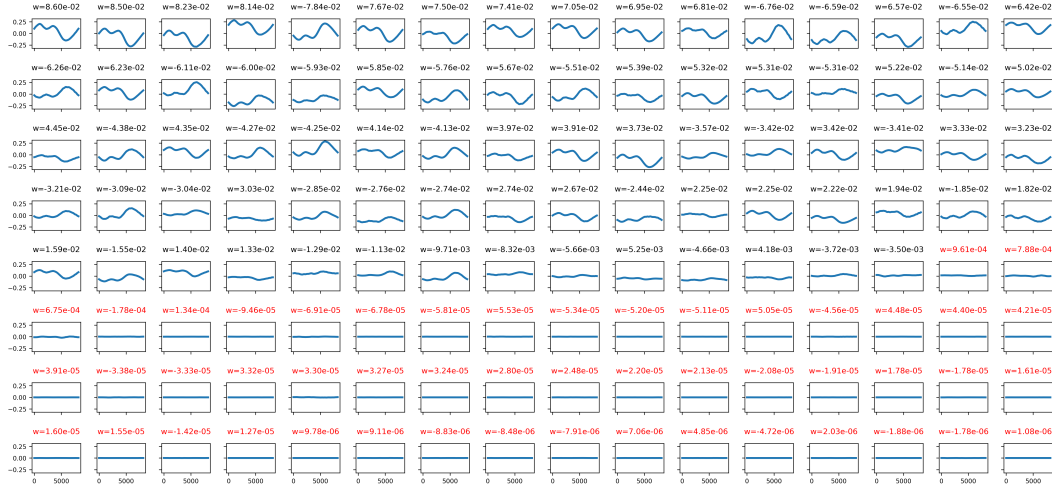


Figure 15: Visualization of the 3D Rayleigh–Taylor instability case. All slices are parallel to the coordinate planes, and their intersection points are chosen as the grid points nearest to the center of the box. We show a representative extreme case, namely predicting the output field after 20 steps from a perturbed initial condition. While other lightweight methods largely fail, SKNO can still capture the overall flow trends and the interfacial surfaces between the fluids with different densities. g represents the gravity direction.

**All 128 Output Pattern fields in the Dictionary
(Sorted by the absolute value of weights)**

FNO



SKNO



Figure 16: Visualization of paired output weights and pattern fields. All pattern fields are sorted in descending order of the absolute value of their associated output weights, and values marked in red lie below the contribution threshold of 10^{-3} . Even with the same sufficiently large N_p , the relative L_2 error of SKNO on the testing data $6.546e-4$ is still larger than FNO $6.603e-4$ (Redundant p -width but still benefit from the basis-diversified evolution).